NATIONAL PUBLICATION FOR BUYING-USING-SELLING COMPUTER HARDWARE & SOFTWARE

APRIL 1986
VOL. 6 NO. 4
ISSUE 75
$2.25
€ 02927
Canada $3.00

COMPUTER SHOPPER™

TERMINAL 141

MONEY SAVING ADS & ARTICLES

Mainframe-On-A-Desk
IBM RT PC
32-BIT SUPER MICRO

In This Issue...

- Hard Disk Backup — Is The Need Being Met?
- Spotlight on Real Estate
- Double Your Memory — Double Your Fun (For Atari 520ST)

JDR 146
B6

See Advertiser's Index Page 6

ATARI ST

Double Your Memory Double Your Fun

by W. Krieger & W. Rostek

When you first get your Atari 520ST, 512K of RAM seems to be more than enough. Think about it, most people are upgrading to an ST from an 8-bit machine with 48K or, in some machines, 64K. Even if the TOS operating system takes up over 200K, that still leaves a lot of free RAM, far more than the 64K you may have been accustomed to. Atari is now shipping TOS in ROM which, when installed, will free up another huge chunk of RAM in the ST.

So, why would anyone in their right mind want to add more memory? Well, one of the truest laws of computers states that there is never too much RAM. The major advantage of having lots of RAM, is in having lots of speed. Access time to acquire data or programs stored in RAM is several orders of magnitude faster than reading that some information from a floppy disk.

Also, programs tend to grow to fill all available memory. In the early days of personal computers, an 8K Basic was a large program that left little room for the application program. Today people are using 64K Basics, and there seems to be no end to the growth. The large amount of memory in modern micros allows programmers to create programs with features and capabilities that would have been impossibe on earlier machines. It seems that the more memory you have in a system, the more you want.

ST owners are very lucky in this regard. We start out with more RAM than most machines, and there is a very inexpensive way to double the existing RAM. The Memory Management Unit, (from here on out referred to as the MMU) on the ST was designed to handle two banks of memory chips. The 520ST only has one bank of chips installed, so all you need to do is install a second bank of dynamic RAM chips, and bingo, you have a one Meg ST.

While this seems simple on paper, actually adding the extra RAM is a little tricky. Atari never designed the 520ST as a one meg machine. **Editor's Note: This was written before the announcement of the 1040ST.** There is no room on the circuit board for the second bank of chips, and there no expansion slots to add more boards. The way to get around this is to piggy back the second bank of chips on top of the existing bank, and fly-wiring the RAS (Row Address Strobe), and CAS (Column Address Strobe), lines to the proper pins on the MMU. All the other signals and power are common to the RAM chips in the original bank of memory.

WARNING

Before going into the step-by-step instructions, we must warn you that this modifications will VOID YOUR WARRANTY.

More importantly, it's very easy to kill your machine if great care is not taken. *Computer Shopper*, or the authors are not responsible for any damage you do. If you don't have extensive soldering experience, don't try to do this mod. We have seen three people kill their STs trying to install the extra memory. In all cases the owners were able to get the machines back in running order, but only after a lot of testing and a lot of extra work. If you do decide to add the extra memory, please read this entire article before you start.

To get the photos and some firsthand instructions on installing the memory mod, we contacted Gary Mckinney of the ST User's Group. Gary has been upgrading STs for User Group members for some time now and has the best track record of successful mods that we have found. His electronics experience is extensive; he has been certified to work on U.S. spacecraft.

O.K., lets get down to the nitty-gritty. You decided you want to do this, and you are confident of your ability to do the job. The tools (See Figure 1) you will need are:

1. A good quality soldering iron with a small tip, preferably temperature controlled. If you can't get a temperature controlled iron, don't use anything larger than a 25 watt iron. A 15 watt iron is better.
2. Thin rosin core solder. The thinner the better.
3. Sixteen 256K 150ns RAM chips. We used NEC 41256C-15s, and had excellent luck with them.
4. An antistatic mat, or a couple of feet of aluminum foil.
5. A desoldering tool or desoldering wick and a solder sucker. The best thing for unsoldering an ST appears to be a desoldering iron from Radio Shack that has a built-in bulb.
6. A couple of feet (4 or 5) of wire-wrap wire. If you can get three different colors, it will help.
7. Phillips screwdriver, long-nosed pliers, wire stripper, tweezers, etc.

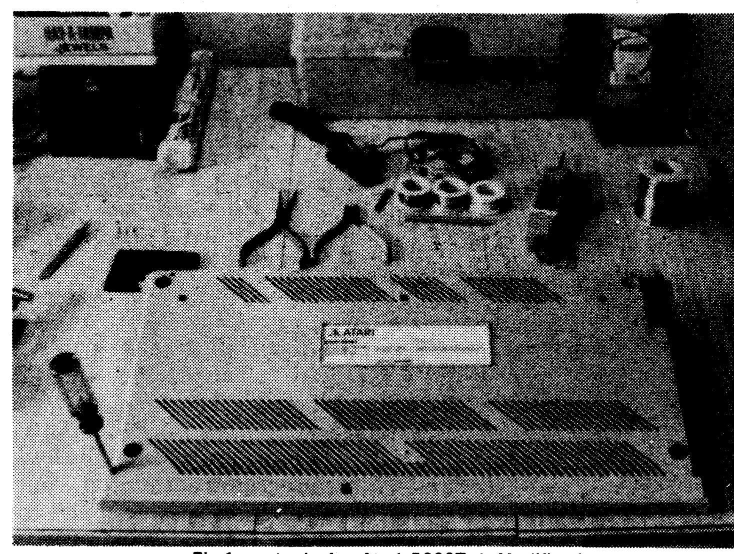


Fig.1 — tools for Atari 520ST & Modification

You don't need to have a logic probe if the modification works on the first try, but if you have problems, it helps to pin down where things went wrong. We recommend that you don't try the mod unless you have a logic probe on hand or can get one if needed.

Step #1

Prepare a clean, well lighted work area. Give yourself plenty of room to work. You will need more light than you normally use because of the close quarters on the ST PC board. Lay out the tools and be sure you have everything before starting. A word of caution about the new RAM chips you just bought, don't take them out of the shipping tubes until you need them. When you do get ready to use them, take proper precautions against static electricity; the 256K RAM chips can be destroyed by static electricity. We have been buying our chips from Microprocessors Unlimited Inc. who

continued to page 142

That Dangerous Thing — A Little Learning

by Frank Tymon

Introduction

A little learning is a dangerous thing -- or so goes the old saying. But there is one thing even more dangerous. That is ignorance. With this routine I'll try to dispel ignorance, hopefully without introducing danger.

The 520ST BASIC program included with this article is a simple example of computer aided learning. The latter consists of a variety of routines, of which this is but one type. This routine assists the user in learning anything of interest to him. He can use pre-generated learning packages, or he can develop and tailor packages of his own.

A picture may be worth a thousand words, but for the computer enthusiasts hands on use ~~is of even more value. In~~ the following paragraphs we'll walk through a typical sequence of devleoping a learning package and using its contents. The pattern can be readily repeated to produce a tool for helping you learn anything from arithmetic to zoology.

TYMON'S TUTOR, A Computer-Assisted Learning Program

Lacking a better name, and being a little short on imagination, I've tagged this masterpiece with the title, Tymon's Tutor. In its own way it does act in the capacity of a private tutor. It teaches you those things you don't know, while giving you the opportunity to periodically review other exercises. As a result you both learn and retain what you have learned.

~~Tymon's~~ Tutor (or TT for short) is a powerful tool for the student, whether in school or learning on his own. Early versions have been used by students throughout the United States, including those in the islands of Hawaii, in academic halls such as Stanford, and in the average high school. It originally ran on the old Trash 80 (A users' term of endearment for the first widely available home computer, the amazingly capable Model 1 Radio Shack TRS-80). A version has been used on the IBM PC. The Jackintosh version takes advantage of the 520ST capabilities to provide an even more powerful, yet user-friendly product.

The Hard Part

The program is rather extensive, and the hardest part in its use is keying it into the 520ST. It is, however, divided into logical segments, and with care they can be correctly entered. Once this hard part is done the rest is gravy. Or, to use some substitute cliches, you are uptown, you're living high on the hog, and from now on its all downhill.

Running Tymon's Tutor

The following steps get you up and running.

1. Bring up your system
2. Load ST-BASIC
3. Enter load "tutor.bas"
4. Enter Run

(The program will set all screens to full, and clear them. At that point it will prompt you for a series of entries leading you to the desired result. The following steps use engrec.dat an English Vocabulary file-to illustrate use of

continued to page 143

The Artistic Atari

by W. Krieger

How many times have you seen a graphics demo on the Atari ST and said to yourself, "Gee, I wish I could do something like that." Well, now that DEGAS is out, you can DEGAS, what's DEGAS? DEGAS is a new program for the Atari ST. It's also an acronym for Design & Entertainment Graphic Arts System; hence the name DEGAS. **(Ed Note: DEGAS is also the name of the impressionist artist.)**

Let me quote Tom Hudson, the program's author, on what the program was designed to do."My purpose in writing DEGAS is to provide an easy-to-use interface that will allow you, the user, to create pictures, diagrams, charts, and so forth with a minimum of effort." Mr. Hudson did a good job, DEGAS is a great program, it's powerful, easy to use, it works in all three resolutions, it's inexpensive, only $39.95 and, best of all it's fun.

Lacking any great artistic talent I decided to first try DEGAS to make up some graphs and charts for a report I was writing. Not bothering to read the manual, I booted up the disk and was presented with a menu of options arranged in columns of boxes, containing lots of interesting things like eraser, draw, point, line, circle, box, text, airbrush etc. But what did he mean by K-line, X-ray-text, mirror, polygon, and a few other cryptic little boxes? Well, if all else fails read the directions.

One of the first things in the manual was a warning to make a back up copy of the disk and put the original in a safe place. In my opinion it's always a good sign when a program isn't copy protected. It shows the company gives some consideration to the user.

The manual is very easy to read and quickly answered my questions as to what all menu options were, how to use them, and gave a few hints and tips on getting the most from the program. All and all, one of the better software manuals I've run into, not too much detail, but enough to answer most of my questions.

The main thing to keep in mind while working with DEGAS is that there are two screens. One is the menu screen where all options such as color, drawing tools, etc. are picked. The other is the working screen. You switch back and forth by clicking the right mouse-button and you must access the menu screen to change any options. This system works a lot better than it sounds. Though it's a bit of a pain to leave the drawing every time you want to change tools or colors. It is really nice to have the whole screen for drawing. The switching to and from the menu screen is very quick and after a short time working with the system it becomes second nature.

All right now that I had a better idea of what was going on it was time to get back to the project at hand. The first thing I wanted to do was to work up a graph. Simple enough, DEGAS has 38 predefined fill patterns that can fill any enclosed section of the drawing, or the whole screen, and another 30 some odd fill patterns supplied as disk files. All I need to do was find a grid pattern and fill the screen, then draw in my axis and plot my points.

Bringing up the menu screen I put the mouse pointer on the fill pattern box and by clicking the left button, cycled through the available patterns. Not seeing a predefined grid that I liked I decided to overlay a pattern of vertical lines with one of horizontal ones. This worked fine and gave me my basic grid to work with.

The other way this could have been done would be to use the design fill option to make my own fill pattern. This opens a window on a 16 by 16 grid, used to make your custom fill pattern. Each square in the grid represents one pixel on the drawing screen. The boxes can either be turned ON or OFF by clicking the mouse when the pointer is in the box. Custom patterns can be saved on disk for future use.

Now that grid was in place, I needed to draw in my axes and label them. Back to the menu screen to select the line option, change the working color from the red used for the grid to black for the axis. And use a thicker brush shape so they stand out. DEGAS has 15 brush shapes to choose from, and you can design custom brushes the same way as fill patterns. The line feature lets you pick the starting point of the line then stretch it out to where it should end then click the mouse and the line is drawn.

If you get the line in the wrong place just hit the UNDO key. UNDO erases the last change you made in the drawing, not just the lines, but any of the drawing options. As long as you haven't switched out the drawing screen between making the mistake and wanting to UNDO it.

DEGAS has excellent text features used to label the axis. The user can choose between different colors, and the sizes of text and the program comes with a custom font editor. This allows you to design any type-font you like. I really like the computer font supplied on the disk, because it prints really well on a dot matrix printer, and without the jagged edges most enlarged fonts give you.

Once the basic background is set up all you need to do is to plot your points, connect them, title the graph, and it's

continued to page 156

Control GEM From Jackintosh BASIC

by Frank Tymon

Jackintosh BASIC's most advanced feature--accessing the GEM graphic capabilities--provides immediate graphic power to the home computer user. The few instructions highlighted in the *ST BASIC Sourcebook* only indicate the potential activities that can be performed. You have seen only the tip of the iceberg. Read on, and you'll realize the extent and value of this option.

Introduction

There are a few instructions (in the text cited above) that arouse your interest in the other possibilities that exist. In the following pages, I'll look at these instructions. *The Sourcebook* provides simple programs to illustrate their use. Unfortunately, these have only meager comments. They allow you to perform a given action, and that is all. I'll explain these routines in depth and show how they can be extended to permit even greater use of GEM.

Additionally, I'll provide some representative programs which exercise GEM through the BASIC interface. However, the programs are only specific examples of extended capabilities. To acquaint you with the alternatives, I'll also develop a matrix of GEM functions against the parameter values that support their use. With this matrix you'll quickly find that drawing a given figure, filling it with a specific color, and modifying it--all using established GEM functions--is a straight forward activity.

There are a few BASIC language reserved words that apply to GEM access. These include gb, vdisys, gemsys, and systab as the primary ones. I'll examine how these allow you to take advantage of GEM, and more specifically the GEM Virtual Device Interface (VDI). This serves as the entry into the GEM graphics.

Well, enough of telling you what I'm going to tell you. Let's move on and find out how to drive GEM, with BASIC language programs.

Doing It

Reading and discussing an idea can be both entertaining and educational. Applying the idea, however, is essential if you want to truly understand the concepts involved. For that reason I'll start off with an actual program. It will draw a simple graphic object and color it internally. The program and a depiction of the drawn object appears on the following pages. Let's examine the code and determine how the picture was drawn. Then we'll analyze the situation and find out how to build on our limited knowledge.

The Circle

For simplicity's sake, let's start off with the circle generator provided in the *520ST BASIC Sourcebook* on page C-156. The program given illustrates what can be done with the GEM system and establishes a pattern that can be extended. I have modified the original program slightly to center the circle on the screen and to enlarge it. See figure 1 and program on page 180.

continued to page 158

Bridge 4.0 For The Atari 520-ST Computer

Artworx has begun its commitment to support the Atari 520-ST computer with the release of BRIDGE 4.0, the popular bridge-playing program that has been available for the IBM, Commodore 64, Atari, and Apple computers. The Atari 520-ST version of BRIDGE 4.0 takes full advantage of that computer's graphics and mouse capabilities. The user never has to touch the keyboard and will find that the cards are displayed just as they would appear on a real bridge table. All bids and card selections are performed by pointing and "clicking" the mouse.

BRIDGE 4.0 for the Atari 520-ST retails for $29.95 and is configured for both the black and white and color monitors.

For further information write Artworx Software Company, Inc., 150 North Main Street, Fairport, NY 14450, or call (800) 828-6573 or (716) 425-2833.

Mention that you read about it in *Computer Shopper*.

ATARI

Applying The Atari

by Jeff Brenner

Still puzzled over how March's *PixMix* program works? This month we'll take a look at the machine language routines essential to this program's operation. We'll also enter a program that gives us a disk-based BASIC language (and fixes up Atari's Revision B BASIC to boot!). And, for all you CX-85 numeric keypad fans, this month's column presents a special keypad driver that lets you use the keypad with some commercial software.

Did You Say 1040ST?

The Atari 1040ST should be in stores by the time you read this. Reaping the benefits of falling RAM chip prices, the 1040ST is an Atari 520ST-compatible computer with 1 megabyte (a megabyte is a million bytes) of memory, standard. Th ebig news is, of course, the price: $999 suggested retail, which includes a monochrome monitor and an internal disk drive.

The 1040ST is essentially a 520ST with an additional 512K RAM. 520ST owners can get equivalent power by getting any of the 512K upgrades that are available (see article in this months *Computer Shopper*.) A few design changes have been made in the 1040ST: The case is deeper, since the disk drive and power supplies are internal. The internal disk drive is double-sided, and an RF modulator is built-in (for using the ST with a standard color television). Additionally, the operating system (TOS) has been completed and (hopefully) fully debugged and is now in ROM. This eliminates much of the boot-up waiting time, since TOS no longer needs to be loaded from disk. The newer 520ST models now contain TOS in ROM also. (Incidentally, the TOS ROM chips are currently being installed at Atari service centers for 520ST owners who presently boot TOS from disks.)

The 1040ST may prove to be the most serious threat to the Apple Macintosh yet. The standard 1M RAM (say goodbye to Ks) combined with a hard disk (to be released soon at under $700) gives this machine enormous potential for serious business applications. Atari may have still even greater plans for reaching the upscale markets; sources say Atari is investigating the user of National Semiconductor's 32-bit "super-chip," the 32332, in a personal computer.

Reader Mail

Dear Jeff,

In the January 1985 column you had a short disk directory program that I've relied on ever since it was published. I put it on all my disks along with an AUTORUN.SYS command that boots the directory program. Eventually I was having trouble on a couple of my disks, because they contained as many as 20 or more short utility files. When the directory program booted and ran on power up, it displayed all the files, but the first few scrolled off the screen (I had to be quick to read them).

To solve this problem, I made a few small changes in your original directory program. Now when the program is booted on power up, or run from BASIC, it displays the disk files in two columns, instead of just on the left hand side of the screen. After it has listed the file names and free sectors, the program asks "RUN AGAIN (Y/N)?" If you press the "Y" key and hit RETURN, the program will prompt you to insert the disk (use only drive one) and press RETURN. When you do this, the screen clears and the program reruns, displaying you files in a double column.

If you answer the "RUN AGAIN" prompt with the "N" key and RETURN, the program erases itself from memory and leaves the file directory on the screen. I find this especially handy as a boot-up program, because it leaves me a list of disk files in plain view to reference when I type in my LOAD, ENTER or RUN command.

Keep up the great work and I hope this program will be of help to your other readers.

Nicholas J. Worth
Heber, AZ

A. I'm sure many readers will appreciate your modification. A holographic sticker is on its way to you for your contribution.

Those readers who want the directory program to RUN when a disk is booted, use the SETUP.COM utility on DOS 2.5 to automatically run a BASIC program utility. See program 1. Next month I'll print a short listing that creates this AUTORUN.SYS file.

Dear Jeff,

One thing I do about *Computer Shopper's* microprint listings is to enlarge them to legible size on an enlarging photocopier (lots of them around.) This also works well for visually impaired folks, by successively copying the enlargement to a size they can read.

Paul Sentner
Pittsburgh, PA

A. Great idea! thanks for sharing it.

Dear Jeff,

We've developed a design for business software that differs from traditional designs in two ways. First, it meets a list of objections we compiled from a survey of operators of small businesses who had investigated computers but were not presently using one. Secondly, because we eliminated many features which were of no value to this group, all design specs are increased dramatically — application speed, functions, flexibility, friendliness, and especially capabity. We developed the design to sell to software developers and manufacturers. Because of it's low cost, availability, and to prove the design would work on any system, we chose the 64K Atari w/1050 drive for our demo system. A byproduct of all this is that we have a program for Atari computers which accomplishes the design goals mentioned. I enclose a booklet documenting the comparative advantages of the MICROMOD program for a user seeking utility from a file manager/word processor. There are not enough of these customers using Ataris to make mass marketing feasible, but we'd like to make it available to those who are.

Steve Boduc
MicroMiser Software
Orlando, FL

A. I'm mentioning your product because of the relative scarcity of true business-dedicated programs available for the Atari. Readers should be aware that MicroMiser calls *MICROMOD* an Integrated Modular Bsuiness System that is not suitable for the hobbyist (as is *HomePak* and *SynFile*), but rather for business persons "seeking utility only." While the specs that MicroMiser sent me look good, I have not seen the actual program in use and thus cannot give an evaluation of its performance. The price is $39.95. Those wanting more information should write to MicroMiser Software, 1635-A Holden Ave., Orlando, FL 32809.

Newsletters

A well designed newsletter was in the mail this month from H.A.C.E., the Houston Atari Computer Enthusiasts (P.O. Box 562, Katy, TX 77492, BBS: 713-644-6400). If your users group publishes a newsletter, send it in and let me know if you participate in a newsletter exchange program. In this way, your group will be able to swap newsletters with other groups reading this column. It's truly interesting to see the Atari newsletters of groups from other parts of the country, Canada and beyond, and I highly recommend that users' groups participate in a newsletter exchange.

Prizes

Cartridge games from Epyx and Sega are being sent to Ricky Freeman, for his Joystick Tester published in the March 1985 issue, and to David Garvin, for his printer utilities published in the December 1985 issue. More prizes will be given away for the best contributions of 1986, and hopefully I'll have even more room this year to print readers' contributions.

For April I have a copy of WSI's 360-page *Atari XL User's Handbook* ($15.95, Weber Systems Incorporated, 8437 Mayfield Road, Cleveland, OH 44026) just waiting to be given away to a lucky reader in our first random (as in RND(0)) drawing. Anyone who writes with a question, comments, contribution, order, etc. before 4/31/86 will automatically be entered in the drawing, or you can simply send me a postcard with your name and address.

New Products

Still looking for a modem? Atari's new XM301 might be just the answer. The XM301 is an auto-dial, auto-answer, 300-baud modem that connects directly to the serial port (or daisy chains with the disk drive) and requires no interface. Included is XE-Term, an excellently designed communications program that supports the 130XE's additional memory. At $49.95 suggested retail, XM301/XE-Term package is a great buy. Contact: Atari Corp., 1196 Borregas Ave., Sunnyvale, CA 94086.

Your Atari 400/800/XL/XE can be just like its big brother ST with a mouse from Zobian Controls. Actually, it's called the "RAT" and includes a mouse-controlled drawing program and a routine that lets your "RAT" move your cursor. A booklet shows you how to use the mouse in your own BASIC programs, but be advised that it won't work with commercial software not designed sepcifically for it. The "RAT" is about $100. Contact: Zobian Controls, P.O. Box 6406, Wyomissing, PA 19610.

AtariWriter Plus Is Here

Has *PaperClip* met its match? Months after Atari's press announcement of *AtariWriter Plus*, I've finally had a chance to try out the new word processor, and my initial impressions are highly positive. *AtariWriter Plus* has all the features that made the original *AtariWriter* popular, plus a lot more: word alphabetization and counting, double column printing (on any printer), "insert" and "typeover" modes, the ability to write a customized printer driver for virtually any printer, an 80-column *editor* (expandable up to 249 columns), a mail merge, and a 36,000-word spelling checker on a separate diskette. A 130XE version of *AtariWriter Plus* is included on the reverse side of the main diskette. The two-diskette package is supported by a 66-page manual. Look for a formal review in an upcoming *Computer Shopper* issue. Contact: Atari Corp., 1196 Boregas Ave., Sunnyvale, CA 94086.

PixMix — How It Works

Last month's feature program, the *PixMix Puzzle*

continued on page 160

Atari Help

by Jeff Brenner

In last month's "Atari Help" column, I listed several manufacturers of memory expansion products for the 800XL. I inadvertently omitted Newell Industries (600 E. Hwy. 78, #602, Wylie, TX 75098). Newell also produces a 256K upgrade for the 800XL ($99.95 complete, or $49.95 without RAM chips). In addition to those readers credited in last month's column for contributing information on the memory expansion topic, let me also thank Danny L. Becker (APO NY) and Karel C. Sysala (Los Angeles, CA), whose letters arrived too late to be icluded in that issue.

XL "Video Smears"

Q. I write in response to the letter in "Atari Help" from Robert Davis of Yorba Linda, California, printed in the December 1985 *Computer Shopper*.

I am having the same trouble using a monitor with my Atari 800XL: namely, a "video smear" such that characters appear blurred. I switched monitors; same effect. Monitor works beautifully on a friend's Apple. Switched cables; no improvement.

Is this an inherent defect in the 800XL series, then? I was able to try a new 130XE and found that characters are much sharper. I was planning to add Omniview to my XL, but am no longer interested: things are bad enough with 40 columns.

I'd appreciate other readers' experiences

Alan McPherron
Pittsburgh, PA

A. Several readers have commented about an apparent blurred Atari 800XL image and in the past I have suggested checking cables, switching monitors, etc. But I am beginning to suspect that the 800XL itself may be at fault. (Apparently, some 800XLs, 1200XLs and 600XLs suffer from the blurred image, but not the 130XE). One theory is that a pixel shift, introduced by the earlier GTIA chip, is responsible for the blurred image. Or had Atari used inferior components in its financially troubled days? An Atari spokesperson said she did not know of any problem. Hopefully we'll hear from other readers on this topic.

Data Files

Q. In experimenting with the DOS files, I tried making a complete record with several strings. What I mean is linking strings together to make one long string: Ex. LET REC$(1,20) = NAME$, LET ADDR$ = REC$(21,40), etc. This seemed to have worked but I got garbage in the blanks. I tried filling the string with blanks before filling it and it still didn't work. How do I accomplish one record from several prompts, and is there a way to view the records in a data file? I've worked with FMS on Data General and we could view the records in the file to verify them.

David Bucci
N. Kingsville, OH

A. You should not get garbage (unwanted data) in the blanks if you properly cleared the main string before assigning the smaller strings to it. I assume that NAME$ and ADDR$ are obtained through normal screen input (i.e. INPUT ADDR$) and are not the source of the garbage. Here's a simple and fast technique that takes advantage of the way Atari BASIC handles its string memory to clear the main string:

REC$=CHR$(32):REC$(MAX)=
CHR$(32):REC$(2)=REC$

where MAX is the total number of characters that REC$ is dimensioned for. The entire string will be filled with character 32, the space character, and subsequent additions to the string will be surrounded by the blank spaces.

To view the records in a data file, you could have your main program read them from disk in the same way they were written (i.e. INPUT #2;TEMP$:PRINT TEMP$;). The maximum number of characters you should read in at one time from a diskette file should not exceed 128 characters. (While up to 256 bytes can be read at a time without an error, anything over 128 bytes will be stored in page 6 of memory, which might upset any machine language routines that you may have stored there.) The best way to do this is to write the main string to disk in chunks of 128 bytes at a time into a temporary string which you can transfer to the main string:

REC$(I,I + 127) = TEMP$.

Another way you can examine a data file is with DOS. Specify the COPY FILE option. When you're asked 'FROM, TO?" type the name of your data file, followed by a comma and an "E:" (i.e. DATAFILE,E:). This will transfer the diskette file to the screen editor.

For a good example of creating and using data file, take a look at the *Mailing List* listing in the June 1985 "Applying The Atari."

Q. A friend has damaged his BASIC cartridge for an Atari 1200XL. Is there any way I can copy the BASIC from my 800XL to disk for him to use. My machine runs the "C" BASIC.

William R. Pew
Everett, WA

A. The program listed under the "DISK-BASED BASIC" heading will allow you to create a diskette which loads the actual BASIC language when booted. It can be used on any 8-bit Atari with a minimum of 48K RAM. (DISK-BASED BASIC will be included on this month's "Applying The Atari" diskette). See program 1.

Type in the program and save it to disk. Place a formatted diskette (containing DOS files) into the drive and run the program. When you press START, the program will write a special AUTORUN. SYS loader to your diskette, and will then save the code for the BASIC ROM in your machine. If you own revision B of Atari BASIC, the program will automatically convert the data into revision C, to eliminate the dreaded revision B bugs.

When it's completed, turn off your computer. Now unplug your BASIC cartridge (or hold down the OPTION key on the 800XL/130XE) and turn the computer on again. The diskette will automatically load the BASIC language into your Atari's RAM, and when the READY prompt appears, you have a RAM-based BASIC.

Once your BASIC is in RAM, you can have some fun with it. Experienced users can examine the memory in locations 40960 through 49151, find where the BASIC commands reside and change them to whatever is desired (the last character of each command must be an inverse video character -- bit 7 set -- and any replaced commands should be the same number of characters in length as the original command). Just be careful where you poke, since you can easily crash the BASIC or make it function improperly. When you run the 'CHANGE THE READY PROMPT" program with your RAM-based BASIC, you can change the familiar "READY" prompt to your nickname, or whatever you want. See program 2.

The "Disk-Based BASIC" loader has been specially designed to permit the SYSTEM RESET key to operate normally. Additionally, DOS 2 and DOS 2.5 users can specify the B. RUN CARTRIDGE option when in DOS and BASIC will be executed, despite the fact that no cartridge is in place!

For those interested in how the RAM BASIC is implemented, two difficulties had to be overcome to make it function properly. First, whenever the RESET key is pressed, a GRAPHICS 0 screen is set up at the top of free RAM. Normally, when the BASIC ROM is in place, the screen is set up directly below the BASIC ROM, at the top of RAM (in systems with 40K or more RAM). But when BASIC is in RAM, this screen is set up on top of part of our BASIC language itself! Unless the disturbed BASIC code is replaced, our BASIC routine is likely to crash.

To solve this problem, a special SYSTEM RESET handler was put into place which automatically replaces 1K of the damaged BASIC code. The result is 1K less of free BASIC RAM (MEMLO is reset upon initialization to provide the 1K of memory for our use) but we are able to use the RESET key normally.

The second difficulty involves DOS. When you call up DOS and choose the "RUN BASIC" option, DOS checks if a cartridge is installed and would normally respond "NO CARTRIDGE" is our RAM-based BASIC is there instead. To get around this, you could press the RESET key to return to BASIC, but if you're using DOS 2 or DOS 2.5, you'll be able to use the "RUN CARTRIDGE" option. An interrupt routine is set up that automatically modifies DOS 2 or DOS 2.5 when it is called up to trick it into believing a cartridge is in place.

The assembly listing of the BASIC loader will be listed here next month.

Requests

If you have been looking for a particular product that others may also be looking for, send me a letter.

Address Atari-related questions to: Jeff Brenner, "Atari Help", c/o *Computer Shopper*, P. O. Box F, Titusville, FL 32781-9990.

DISK-BASED BASIC

```
YCJ 10 REM DISK-BASED BASIC INITIALIZATION
KGJ 20 REM COPYRIGHT 1986 JEFF BRENNER
RDJ 30 DIM BASIC$(8192),S$(35),CIO$(7):BASIC$(8192)=CHR$(0)
KFJ 40 FOR I=1 TO 7:READ N:CIO$(I,I)=CHR$(N):NEXT I
GXJ 50 FOR I=1 TO 35:READ N:S$(I,I)=CHR$(N):NEXT I
QVJ 60 PRINT CHR$(125);"THIS PROGRAM CREATES AN AUTORUN.SYS"
NXJ 70 PRINT "FILE ON THE DISK, ALONG WITH A FILE,"
XFJ 80 PRINT CHR$(34);"BASIC";CHR$(34);", WHICH CONTAINS THE BASIC"
QEJ 90 PRINT "LANGUAGE.":PRINT :PRINT "INSERT BLANK FORMATTED DISK INTO DRIVE"
XTJ 100 POKE 85,7:PRINT "PRESS <START> TO CREATE FILE"
PCJ 110 IF PEEK(53279)<>6 THEN 110
HJJ 120 PRINT :PRINT "CREATING AUTORUN.SYS..."
THJ 130 CLOSE #1:OPEN #3,8,0,"D:AUTORUN.SYS"
WRJ 140 RESTORE 240:FOR I=1 TO 309:READ N:PUT #3,N:NEXT I:CLOSE #3
JGJ 150 A=USR(ADR(S$),ADR(BASIC$)):IF ASC(BASIC$(2272))=202 THEN GOSUB 500
MUJ 160 POKE 881,0:OPEN #3,8,0,"D:BASIC":PRINT "SAVING BASIC..."
RSJ 170 H=INT((ADR(BASIC$)-6)/256):L=ADR(BASIC$)-6-H*256:POKE 884,L:POKE 885,H
GOJ 180 POKE 888,0:POKE 889,32:POKE 882,11:I=USR(ADR(CIO$),48)
HQJ 190 CLOSE #3:PRINT "FINISHED.":END
QYJ 200 DATA 104,104,104,170,76,86,228
EEJ 210 DATA 104,104,133,205,104,133,204,169,0,133,206,169
EMJ 220 DATA 160,133,207,160,0,177,206,145,204,200,208,249
YOJ 230 DATA 230,205,230,207,166,207,224,192,208,239,96
UPJ 240 DATA 255,255,215,5,255,6,165,12,141,108,6,165
UGJ 250 DATA 13,141,109,6,169,107,133,12,169,6,133,13
UHJ 260 DATA 162,0,142,68,2,232,134,9,169,160,133,106
OLJ 270 DATA 32,175,6,78,197,2,160,3,140,114,3,200
VZJ 280 DATA 140,122,3,169,220,141,116,3,169,6,141,117
RIJ 290 DATA 3,162,48,32,86,228,169,7,141,114,3,169
VTJ 300 DATA 250,141,116,3,169,159,141,117,3,206,121,3
UUJ 310 DATA 32,86,228,14,197,2,169,119,141,252,2,173
NVJ 320 DATA 231,2,133,2,141,229,6,173,232,2,133,3
TPJ 330 DATA 141,230,6,24,105,4,141,232,2,169,214,141
WIJ 340 DATA 250,191,169,6,141,251,191,162,4,160,0,185
QSJ 350 DATA 0,188,145,2,200,208,248,230,3,238,86,6
URJ 360 DATA 202,208,240,169,188,141,86,6,208,44,32,0
QEJ 370 DATA 0,169,160,133,106,32,175,6,32,200,6,24
OAJ 380 DATA 105,4,141,232,2,162,4,160,0,177,2,153
WGJ 390 DATA 0,188,200,208,248,230,3,238,134,6,202,208
UMJ 400 DATA 240,169,188,141,134,6,173,34,2,141,254,6
QOJ 410 DATA 173,35,2,141,255,6,162,6,160,231,169,6
PHJ 420 DATA 32,92,228,76,0,160,169,3,141,98,3,169
TTJ 430 DATA 228,141,100,3,169,6,141,101,3,169,12,141
QYJ 440 DATA 106,3,162,32,76,86,228,173,229,6,133,2
OBJ 450 DATA 141,231,2,173,230,6,133,3,96,32,200,6
MRJ 460 DATA 76,80,6,68,58,66,65,83,73,67,155,83
SUJ 470 DATA 0,0,173,249,38,201,253,208,5,206,249,38
VGJ 480 DATA 208,10,173,89,39,201,253,208,3,206,89,39
OQJ 490 DATA 76,0,0,224,2,225,2,215,5
MZJ 500 PRINT "CONVERTING REV.B TO REV.C BASIC..."
BRJ 510 BASIC$(2272)=CHR$(234):BASIC$(2273)=CHR$(240):BASIC$(2274)=CHR$(17)
AQJ 520 BASIC$(2275)=CHR$(234):BASIC$(6954)=CHR$(0)
DCZ 530 FOR J=8180 TO 8186:BASIC$(J)=CHR$(0):NEXT J:RETURN
```

PROGRAM 1

CHANGE THE "READY" PROMPT

```
ELJ 10 REM CHANGE THE "READY" PROMPT
RVJ 20 M=48504:IF PEEK(48498)=169 THEN M=48493
YDJ 30 DIM A$(6)
GAJ 40 PRINT "WHAT'S THE NEW PROMPT";:INPUT A$
NMJ 50 IF LEN(A$)>5 THEN PRINT "5 CHARACTERS MAXIMUM":GOTO 40
UUJ 60 IF LEN(A$)<5 THEN FOR I=LEN(A$)+1 TO 5:A$(I,I)=CHR$(32):NEXT I
CBJ 70 FOR I=1 TO 5:POKE M-I,ASC(A$(I,I)):NEXT I
MAZ 80 PRINT "PROMPT HAS BEEN CHANGED!"
```

PROGRAM 2

Double Your Memory continued from page 43

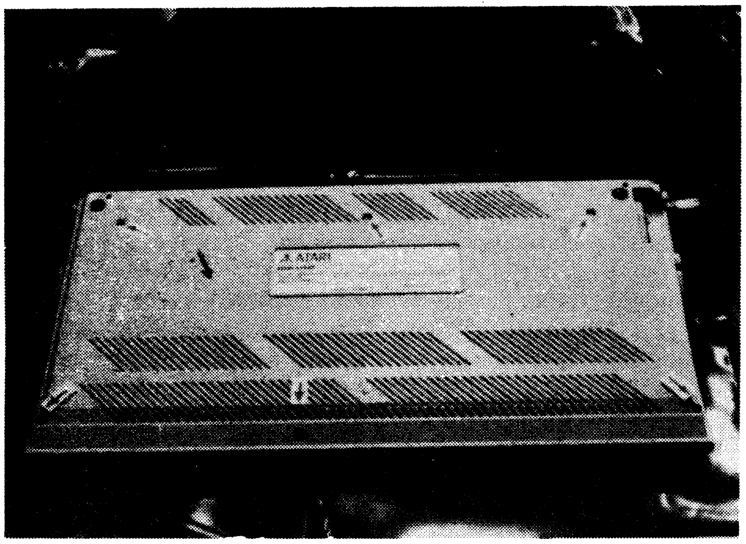


Fig. 2 — Screw holes for disassembly

provide a sheet with more information on static electricity and RAM chips than you would believe.

Step #2

Turn your ST over and remove the six Phillips head screws (See Figure 2). Notice the longer screws all came out of the rear holes. If during reassembly you use the long screws in the front they will extend through the top of the case. Remove the top of the cover, and lay the keyboard to one side (See Figure 3). Note the orientation of the keyboard connector while you unplug the keyboard; one pin on the connector has been keyed. Now, unscrew the circuit board from the bottom of the cover and remove it from the plastic case. Lift the front edge of the board/shield assembly, and wiggle it back and forth as you pull it towards the front. Don't force it; they tend to bind in the back where the connectors poke through the case. Wiggle gently.

Now, remove the RF shielding from the board. The two sections are held together by tabs (See Figure 4) that extend from the bottom section through the PC board and are bent over to hold the top in place. On some early machines, some of the tabs of the RF shield will be soldered. These must be unsoldered.

Step #3

Start taking precautions. With the PC board removed from the RF shield, it's easy to locate the RAM chips and the MMU. The RAM chips run in a straight row along the right front of the board. The MMU is directly above the leftmost RAM chip (See Figure 5).

Lay down your anti-static mat. If you don't have one, a couple of sheets of aluminum foil work just as well. Note the alignment of the MMU chip. There is a small depression in the surface that indicates pin 1. The pin 1 side goes towards the bottom of the board. Make sure you remember how the chip goes in the socket; if you put it back wrong, it will burn up.

Gently remove the MMU chip from its socket by prying under the corners of the chip with the end of the tweezers. Set the MMU down on a square of aluminum foil, wrap it up in the foil, and set it aside in a safe place.

Warning

Do not apply power to the machine from this point on. If you apply power before the complete modification is installed, you will damage the machine.

To the left of each RAM chip is a capacitor that must be removed (See Figure 6)to give you working room. Desolder and remove them. Take your time, and don't force them out. The traces on the PC board are very, very thin and easy to destroy. Too much heat will lift them right off the board. If you pull on the capacitors to get them out, be very gentle; the traces pull off with the slightest pressure.

Once all the capacitors are out, go back and clean up the

continued on page 144

Silver Box
continued from page 140

MSDOS 1.25 or on any other version of MSDOS other than 2.11, or on any computer other than the Sanyo MBC-550/555 -2. A general MSDOS program SH80 is being tested for release in mid 1986.

(Author's Note: I could not finish the MIDI experiments in time for printing, but the project is in progress. Do you have any suggestions on this project?)

Q And A

I received a few questions asking what makes the Sanyo MBC-550/555 different from the IBM PC and Clones.

The biggest difference is in the video display system Sanyo uses and the Disk Controller chip Sanyo has employed. The Sanyo video display consists of three video memories for Red, Blue and Yellow, each 16K long. Each bit in the memory represents a tiny dot on the screen arranged as pixels of 640 X 200. The Sanyo screen has the same resolution as in the IBM monochrome mode but in **color**! Each dot from three 16K memories are super-imposed on the screen to produce different colors.

The output of each memory is independent and separately output as Red, Green and Blue (RGB). Actually the screen phosphor colors are Magenta, Yellow and Cyan, not RGB, in RGB monitors. One character is eight dots wide and eight dots high. One horizontal character line is divided into two lines. In other words, each line is composed of one byte wide (eight dots) and four bytes high. In order to compose one character, you need two lines comprised of four raster lines each.

You can experiment this by using Debug. After starting the Debug, type E3C00:0600 or EF000:0600 or EF000:4600, using MSDOS 1.25 or 2.11 but not the so called Color Dos, fill each position by typing characters representing the HEX values. This will place the dots directly on the screen bypassing the BIOS. Experiment with FF (eight dots), AA (four dots in positions: D7, D5, D3 and D1) or 55 (four dots in positions: D6, D4, D3, D0). (More on the Sanyo difference will follow.) ●

That Dangerous Thing
continued from page 43

TT. Not to worry where it came from, we'll soon generate it!)

5. Enter engrec.dat

6. Enter 3, to read OUt of the engrec.dat file
(if you were developing or tailoring a file, or adding to one, then you would enter 2, to read INto the file being used. For CORrecting an existing file you would type in 1.

7. Enter 1
(New entries are keyed as 1. As you study the material, and successfully relate the English word with the displayed meaning, the key is increased incrementally and automatically. A review program in which you review 1-key entries daily, 2-key words weekly, 3-key words monthly, 4-key words quarterly, 5-key words semiannually, and 6-key words annually has advantages. You don't waste time learning what you already know. Yet you review often enough to retain a high percentage of the given vocabulary. Here we are using English vocabulary but any material to be learned can be used. That is, TT allows you to tailor your own education package.)

8 Respond to the displayed meaning with the appropriate word

9 repeat until no words remain, or

9.1 enter LAST to end the session

Developing A Learning Package

1 Enter the system
2 Load ST-BASIC
3 Enter load "tutor.bas"
4 Enter Run
5 Enter a file name. (For example, engrec.dat)
6 Enter 2 (for entering new data INto engrec.dat)
7 Enter 1, required response, cue to be displayed by the computer
(for example, sticking with our engrec.dat file, you might enter

1,cugnet,swan

8 Repeat for each additional learning element
9 Enter last,a,a to quit this session (Note that you can return to your data file, here engrec.dat, in subsequent sessions and add more entries)

Adding Entries To An Existing Learning Package

1 Bring up the system
2 Load ST-BASIC
3 Enter load filename (for example, load engrec.dat)
4 Enter 2
5 Enter 1,required response, cue to be displayed by the computer
(for example, staying with our engrec.dat file, you might add

1,cozy,comfortable

6 Repeat for each needed entry
7 Enter last,a,a to quit this session

Correcting Entries

1 Bring up the system
2 Load ST-BASIC
3 Enter load filename (for example, load engrec.dat)
4 Enter 1
5 Enter required response (such as cozy) to access record to be corrected
6 Enter 3 to change required response field
7 Enter new entry, for example, relaxed
8 Enter 4 to change cue field
9 Enter new cue entry
10 Enter 1 to repeat the process - records are sequential
11 Enter 2 to finish or end the process

General

The above entries, as well as one for displaying all learning records, are prompted by the program itself. For example, to see all learning records, when quieried print SHOWME. Otherwise merely strike return.

This is the essence of the functions performed by TYMON'S TUTOR. The provided record size is 68 characters. The actual entries are limited to 15 characters for the required response, and 45 characters for the displayed

continued on page 194

Double Your Memory continued from page 142

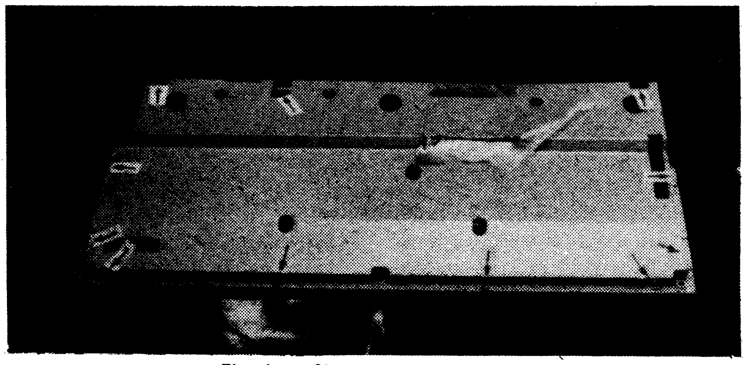


Fig. 4 — Chassis tabs locations

holes. They will be difficult to get to when the second bank of RAM is in place, so clean them now.

Step #4

Prepare your new RAM chips. First, the chip legs are slightly bowed (splayed). You must remove the splay from the legs by bending all the legs in slightly. The easiest way to do this is to hold the chip with needle-nose pliers against the table so that all the legs on one side are lying flat on the table, then rotate the pliers slightly. Turn the chip over, and bend the legs on the other side in. The legs should be perpendicular to the IC body now. (See Figure 7)

Next bend up the RAS and CAS pins (Pins 4 and 15). Bend the entire pin so that it comes straight out of the IC. Next, bend the thin portion of the leg so that it points straight up (See Figure 8).

Slide the new chip over the top of the existing one. Make sure that all pins fit tightly and that the notch is on the same side on both chips. Examine the alignment of the pins on the top and bottom chips. It is very easy to create a solder bridge between pins, so it is important that the pins on the top and bottom chip align (See Figure 9). When you are satisfied that the chip is positioned correctly, solder the legs of the new chip to the corresponding leg on the existing one. Work one side of the chip at a time and be extremely careful to get a good connection. It is easy to flow solder on the top pin without actually soldering the connection when you are working in such a tight place. Be on the lookout for solder bridges; they are very easy to create.

Step #5

Reinstall the capacitors that were removed in step 3. Be sure to bend the legs of the capacitors before soldering so that they don't brush up against the RF shield when it is installed. This is your last

continued on page 151

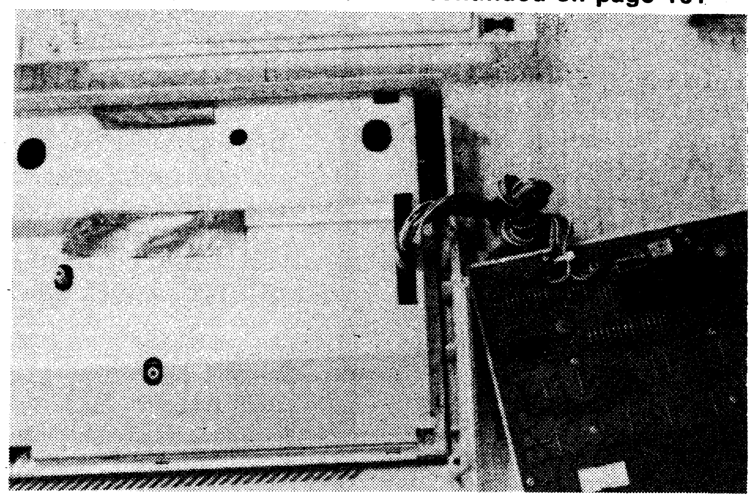


Fig. 3 — Keyboard and Connector

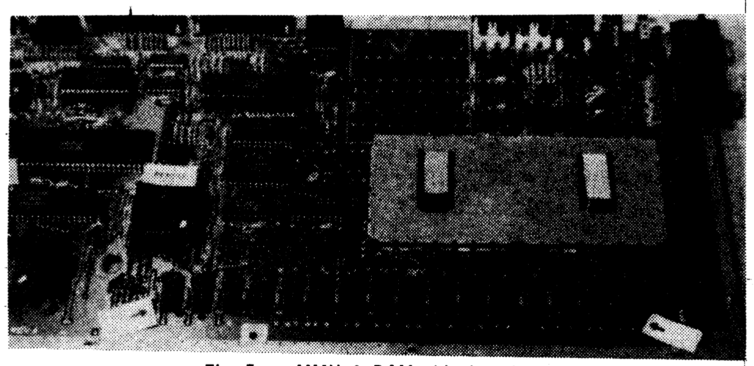


Fig. 5 — MMU & RAM chip locations

Double Your Memory continued from page 144

good chance to look for solder bridges and poor solder joints.

Step #6
Warning

Never apply power when any RAM chip has a pin that is not connected. You will destroy the RAM chip.

In this step run the three fly wires to bring the RAS and CAS signals to the new memory bank. The first wire connects pin 4 (RAS) (See Figure 10) of all the new chips to pin 18 of the MMU. To do this, measure out a length of wire along the RAM bank that's about 4 or 5 inches longer than that of the PC board.

Cut three wires this length. If you have three different colors, the job is a little easier; but, it is not vital.

Note!

Be sure to keep your connections as close to the IC as possible, because you will need to trim the bent up pins as low as possible so that they don't touch the RF shield when it is replaced.

Strip the insulation off the first wire so that the bare portion of the wire is two inches longer than the length of RAM array. Position the wire so that the insulated portion of the wire is to the left of the RAM chips and the bare portion lies across the RAM chips. Starting at the leftmost RAM chip, (U16) wrap a loop around pin 4, and solder the wire to the pin. Continue looping and soldering until all the chips are connected (See Figure 10). Clip off and discard the remaining portion of bare wire.

Take the second and third

continued on page 154

Figure 7

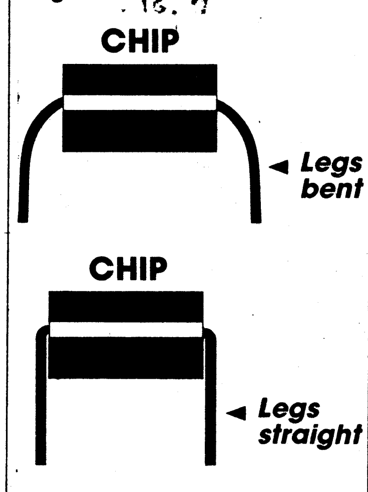
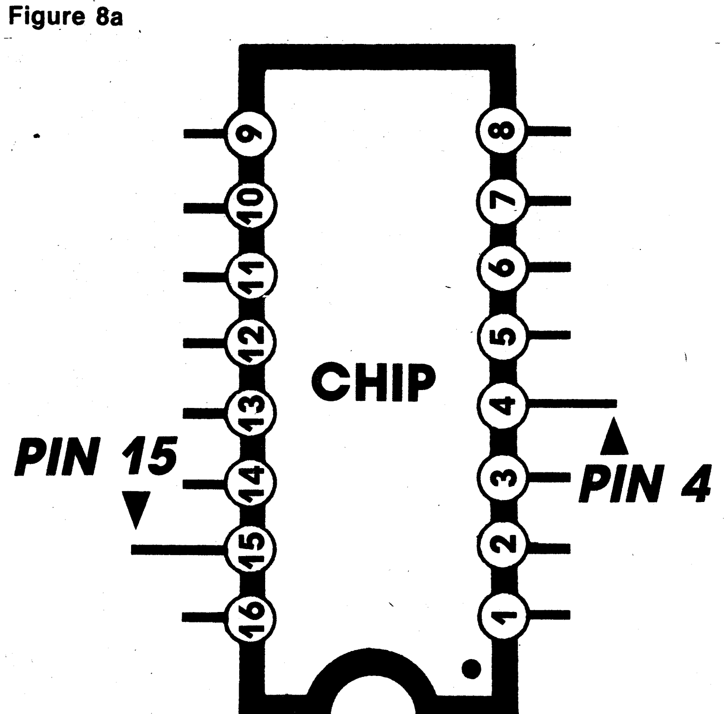


Figure 8a



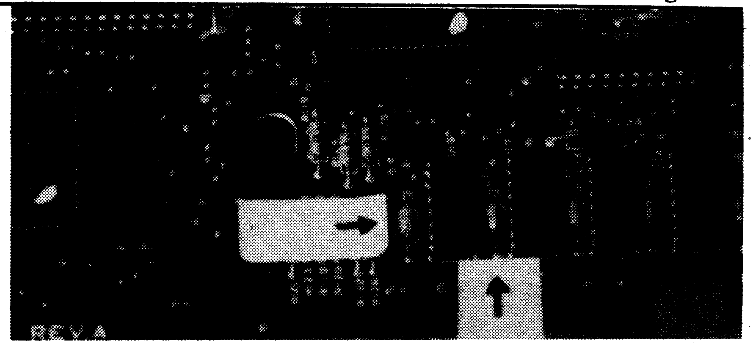


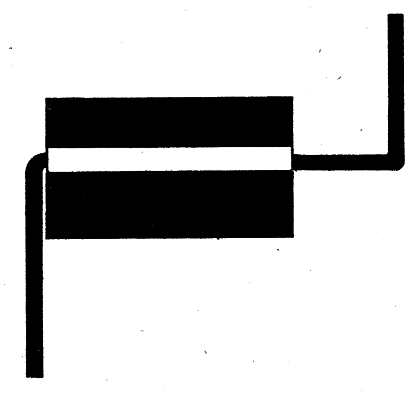
Fig. 6 — Capacitors to be removed before installing chips

Double Your Memory
continued from page 151

wires and strip enough insulation to connect eight chips. (Remember to strip 2 inches extra.) Start with the second wire, and the leftmost chip, (U16) and loop the wire around pin 15. Solder the wire and continue looping and soldering until you have connected the left eight chips. Clip off the excess wire and discard it.

Using the third wire, start at pin 15 of U32, and wire pin 15 of the last eight chips together.

Figure 8b



Note that the MMU chip pin numbering doesn't have pin 1 on a corner as you would expect. The MMU pin numbers are silk-screened on the component side of the board. Pin 1 is in the middle of the lower edge. Turn the board over. (See Figure 11 to help locate the pin locations on the bottom side.) Be very careful. You will also notice that the MMU socket has two rows of pins on the solder side of the board.

Since pin 1 starts at the middle of the row, the last pin of the outer row of lower pins is pin 9. You will notice that just around the corner of the socket is pin 10. As you turn the corner, the even pins become the outer row and the odd pins become the inner row. Refer to Figure 11 to help you identify the correct pins.

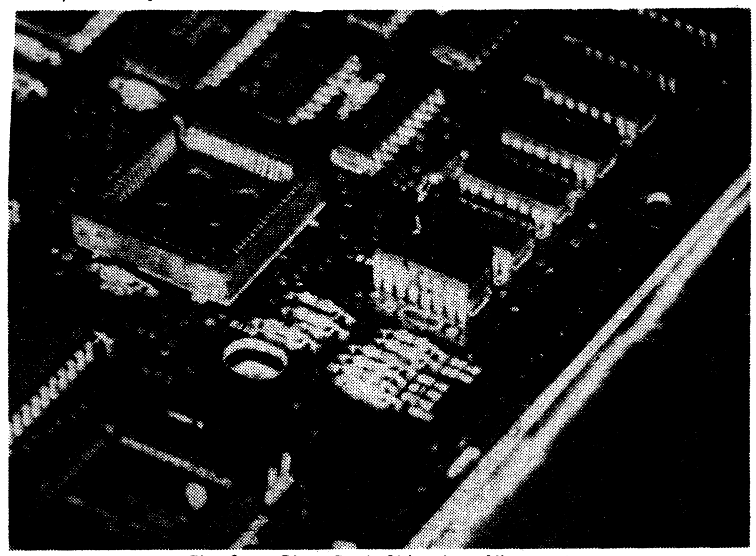


Fig. 9 — Piggy-Back Chips Leg Alignment

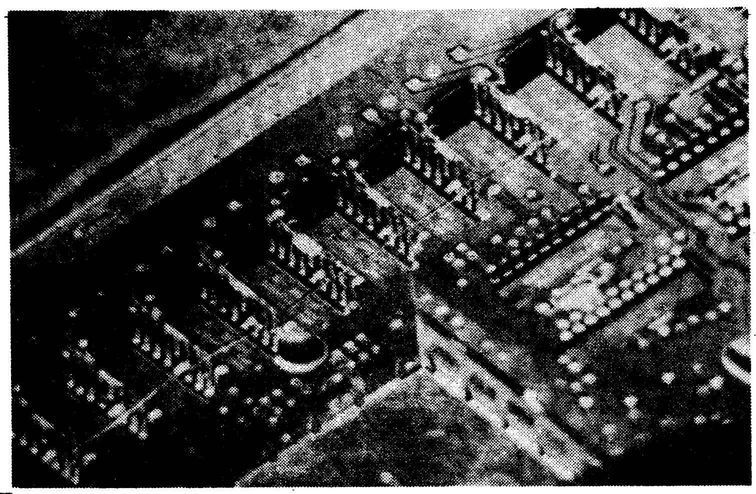


Fig. 10 — Fly Wire installed to Pin 4

Route the three wires through the hole below and to the left of the MMU. One at a time, measure out the wires to the proper pins on the MMU, trim them to length, and solder them starting with wire 1, which goes to pin 18 on the MMU. Be careful when soldering to the MMU socket pins; they can move around a great deal if you push them while the solder is molten.

The second wire goes to pin 21 of the MMU socket. The third wire is connected to pin 22 of the MMU socket. Now that all of the wires have been soldered, replace the MMU chip.

For all practical purposes the job is now done. Take a good look at your work and be as sure as possible that everything is right before you turn the power back on. Keep an eye open for solder bridges. Don't bother to reassemble the whole thing yet. Place the PC board in the bottom half of the case. Plug in the keyboard and hook up the system to your monitor, drive, and power supply.

Turn on the power. If it boots, you should be all right. If the system boots, try to copy a complete disk to a second disk by dragging the icons on top of one another. You should be able to copy the complete disk in a single pass. If the system doesn't come up at all, the odds are good that have the MMU in wrong (**in which case it is dead**) or that you killed a RAM chip with static. If you can copy a disk, but not in one pass, odds are good that there is a pin that looks soldered, but isn't.

If your system doesn't work, there are alternatives to suicide. Look at the pins of the memory chips with the logic probe. Notice a pattern that appears on all but the power lines of the ICs. By looking at each pin of each IC, you should be able to find the IC that has the abnormal pattern. Once you have located the pin, you must figure out why the signal isn't there. Look for a pin that isn't soldered or for a bridge. Be on the look out for a crack in the traces on the PC board; we have seen this happen even though we didn't go near them for the modification.

Now that you're happy that everything is working, unplug everything and reassemble the machine. Reassemble the bottom RF shield to the PC board. (Or throw it away if you are worried about heat build up.) Replace the screws that secure the PC board to the bottom case, and install the upper RF

continued on page 156

The Artistic Atari
continued from page 44

done. However, instead of plotting all the points and then going back to draw in the lines, lets go ahead and use the K-line, or continuous line feature. This works the same as the standard line function except that the line doesn't stop when you click the left button, instead it continues on to the next point you stop the mouse pointer at and click the left button. Perfect for drawing graphs.

DEGAS also has a bunch of different line patterns although they weren't needed for the graph I was working on, and they are just great for doing multiple plots on the same axis. A good example would be comparing different year's sales figures, for example, 83 might be a dashed line while 84 can make up dots and 85 as a solid line.

Once the graph was done, I wanted to get rid of the grid I had set up for plotting my points. Again it was a simple thing to do with DEGAS. All that was needed was to overlay the grid fill pattern with a blank fill pattern. You can think of it as writing over the grid lines with the background color and effectively erasing them. The only problem with this is that it left little sections of the grid in the lettering in the title and on the axis. I first tried to "fix" this by carefully point drawing with the background color. This didn't work very well, I couldn't control the mouse finely enough not to mess up my letters.

Luckily I had taken the manual's advice and saved a copy of my work to disk before I tried to – fix" it. What was needed to clean up the problem was to use my magnify mode. This takes a 20 by 40 pixel section of the work and expands it to fill the entire screen. Then you can work with individual pixels with ease.

Now that everything has cleaned up to my satisfaction it was a simple matter to dump the graph to my printer using the print pic option. Currently the only printer driver is for Epson compatible printers. The manual promises that other printer driver programs will be ready soon and they will be made available in later versions of the program and possibly on Compuserve's SIG-ATARI. Registered owners will be advised when other disks are available. This will be great, since DEGAS really needs to be able to dump to a color printer.

This recap of my first session with DEGAS just hints at all the program's features, since I didn't even use some of the most powerful features. For example, the shadow option creates a shadow of border around any figure, angled in the direction of your choice, in the color you desire, and at a distance you specify. This option makes 3-D effects really easy, both on drawings and lettering.

DEGAS also creates circles, boxes, frames or disks, by having the user selecting only two points, the programs draw in the rest. For example, to draw a circle you only need to click the mouse on the center of where you want the circle. Then move the pointer outward from the center to define the radius. When the circle is the right size, click again and DEGAS draws it in.

I used the circle, shadow, and ray features to make some really nice 3-D pie charts. Rays work like other lines but they all have a common point of origin. These make pie charts like the ones I've tried, easy to do and they look great. I also used the polygon feature to do some bar graphs.

I've mainly stressed DEGAS business graphics capabilities because that's what I am most familiar with. But DEGAS is much more flexible, some of the art work that I've seen created with it is stunning. There are still a few features of DEGAS that I haven't mentioned, let alone explained in detail. Let me just say that half of the fun I am having with this program is in finding out what all I can do with it.

Not many of the software tools I use are really a lot of fun. Oh sure they do a great job, making my work easier and faster, and with fewer errors. I also have many games for my ST that are a lot of fun, but it's really nice to be able to do some productive work on my system and have a great time while I'm doing it. I really like this program, and recommend it to anyone who wants to try his or her hand at computer graphics. ●

Double Your Memory
continued from page 154

shield. Plug the keyboard in, put on the upper case, and replace the case screws. Remember that the long screws go in the holes along the rear edge of the case.

How do you get the best use of all this extra memory? We recommend a RAM disk program. This is a nice little program that takes a section of your RAM and makes it look like a disk drive to TOS.

You won't believe the performance improvement some programs show unless you have seen a RAM disk in action before. For example our compiler runs at least 20 times faster from the RAM disk than it did off the floppies.

We have seen two RAM disk programs; one is in the public domain and can be obtained from a users group or downloaded from SIG ATARI on compuserve. The problem with this program is that it works only on one meg machines and creates a fixed sized RAM disk (the same size as a single sided drive). The other one we have seen is the one we use. M-Disk from Michtron is our recommendation if you are in the market for a RAM disk. It allows you to create any size disk you want, and it will work on a standard ST. (But you can only create a RAM disk of about 120K unless you have done the memory upgrade.) M-Disk is $39.95 and is available in many computer stores.

If you wish to have your machine upgraded, but don't feel like doing the job yourself, contact the ST Users Group BBS at (305) 383-1413 for details on having Gary install the mod. ●

MMU PIN NUMBERING

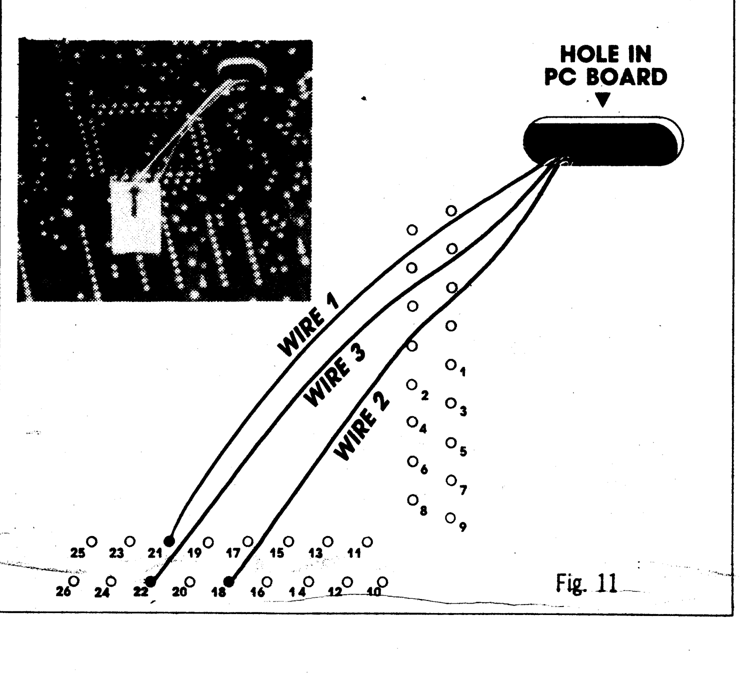


Fig. 11

Control GEM
continued from page 44

The first two lines identify what the program is doing, set the color parameters, and provide for a full output screen. The subsequent lines are the keys to entry into the control of the GEM's VDI. In line 30 the entry 11 tells the system that a circle is desired. The following poke control lines, then specify the parameters VDI associates with circle generation. After this the ptsin lines establish the X and Y coordinates of the center and the radius of the circle. Finally, vdisys(1) triggers actual execution of the VDI function.

It is evident that the entries through line 60 do all the preliminary work for generating our circle. Lines 70-90 establish the essential information on location and size, and line 100 implements the job of drawing the circle. For practical purposes, we then can draw a variety of circles by merely modifying our entries in lines 70-90. To convince yourself of this, I recommend that you enter the above program and then experiment with various locations and sizes of circles. (You can, of course, change the color parameters if you so desire.)

And An Ellipse

Well, if that were the limit of our capabilities, we would soon tire of the computer. Of course, we can draw a circle without using the VDI option. So, what else can this facility do for us?

Actually, it can do a great deal more than generate circles. But, that is an easy exercise to start with, and it provides the building blocks for other activities. For example, let's examine how we might draw an ellipse. See figure 2 and program on page 180.

Wallah! Not much different in appearance is it? Evidently the same basic Contrl parameter, 11, covers both circles and ellipses. However the parameters aligned with contrl + 2 and contrl + 10 have changed. These indicate to VDI that an ellipse is needed. And, although we locate the center as before, we now need to provide an additional radius to generate the elliptical shape. Otherwise the same basic pattern prevails.

As before, do generate a few ellipses scattered hither and yon over your screen, using different sizes. Lines 70-100 are the only ones that require changing.

A variant on this theme is possible with only a slight change in the program. We'll make this change and then produce a partial ellipse defined by starting and ending angle values. In providing the information to the VDI, the angles must be in such units that 3600 constitutes a complete circle. The entry in intin is the starting angle (0), and the value in intin + 2 is the ending angle (2400). Program 3 produces a pie that is roughly three quarters of an ellipse.

Old-Fashioned Straight Lines

A pattern for operating with the VDI is beginning to shape up. I'll return to this pattern and provide a simple matrix that correlates the desired graphic product and the appropriate parameters. First, though, we'll examine one more program that differs a little more drastically from the examples on page 180. See program 4 and figure 4 on page 180.

In this routine, we draw a sequence of straight lines between specified points. Again, there is a BASIC instruction that can also draw lines. But our concern here is less with the line than with the VDI implementation. When 6 is assigned to contrl it recognizes that a line is to be drawn, and a 3 in contrl + 2 confirms this. At this point, the VDI system knows that the ptsin entries will be X and Y coordinates.

Of course, we need not draw each of the objects above with individual programs. They can be integrated into a single program that will, for example, impose a circle on the ellipse, and an X-Y axis on the screen. The result is shown in program 5 and figure 5 on page 180.

As you can see, program 5 doesn't really introduce much that is new. Rather it chains together the ellipse, circle, and polyline into a simple routine. The polyline portion differs from the earlier one in that some of the lines overlap each other, but the procedure is the same as before. Admittedly, these particular routines can be emulated by some of the BASIC instructions provided. Nevertheless, they highlight the structure for accessing the powerful GEM VDI capabilities.

We Bar Nothing In Our Program

One additional illustration of the VDI capability is of interest. That is the bar chart. Here we still use the control 11, etc., used with the circle and ellipse. Contrl + 2 is set to 2. Contrl + 10 is set to 1, however. Then, in ptsin and ptsin + 2, we set one corner of our bar. The diagonal corner opposite this is in ptsin + 4 and ptsin + 6. That's it. Short, simple, and to the point. In our routine, we draw both a horizontal and a vertical bar. See program 6 and figure 6 on page 180.

From these programs a pattern for VDI usage should now be recognizable. Can we, however, build on this pattern? Certainly we can. All we need is information on the appropriate parameters to be used with the routine. Not to hold you in suspense, the table on page 180 constitutes a summary of such parameters.

VDI Parameters

VDI uses a set of Operation Codes (Op Codes) to specify the function to be performed. These codes range from 1 to 131. We have been using them in the simple programs presented so far in this article. For example, when we assigned the value 11 to our variable, contrl, we were specifying the op code for the VDI Generalized Drawing Primitive (GDP).

Within the VDI construct there is a secondary set of parameters called Ids. For the GDP, there are ten such Ids, each producing a different drawing. For example, Id 1 is used to identify that function that draws a bar. Id 2 is used for drawing an arc. Id 3 draws a pie shape. And so on for other shapes.

We specify these parameters by assigning them to contrl + 10. If contrl + 10 is set equal to 9, we would be able to draw a rectangle, with its surface filled with a selected colored pattern and its corners rounded. What do the other contrl values signify?

Contrl + 2 contains the designator for the number of vertices to be associated with the ptsin entries. Contrl + 6 establishes the length of the input array, intin.

These few contrl parameters allow us to designate what we want the GDP function to do for us. You might note that the initial contrl value is a reference address. Two bytes above this is the next contrl value. Two bytes above this is the next contrl value. Two bytes above this is the next, and so on. In other words, for the contrl, as well as for other variables we use, we are merely indicating the value to be input at an address displaced from the initial one assigned. Thus, when you set ptsin to a value, the value is assigned to an address. The value assigned to ptsin + 2 is assigned to an address two bytes away. And similarly for other entries.

The ptsin values are used to, among other purposes, specify coordinates. In the circle routine, they indicate the center of the circle. For a rectangle, they would indicate the initial corner and that corner opposite. Ptsin could be the initial x-value; ptsin + 2, the y value for the initial corner. Ptsin + 4 would be the x-value

continued on page 160

Applying The Atari
continued from page 45

Machine, was built around five machine language routines that enable the program to function quickly and effectively. The assembly listings of these routines are being listed this month for those of you who wish to study them, or to incorporate them into your own programs.

When *PixMix* loads the digitized watch picture (or any screen) from disk or cassette, it stores the data in a string (SCR$), rather than directly in screen memory. This allows the program to have continuous access to the original picture data, regardless of what is done to the picture on the screen (i.e. scrambling). Once the data is loaded into the string, the "STRING TO SCREEN" routine is used to transfer the data to the screen, leaving the contents of the string intact. See program 2.

The "GET SCREEN PIECE" and "PUT SCREEN PIECE" routines allow high-speed manipulation of the graphics screen. The "GET" routine picks up a 64 by 32 pixel area and stores it in P$. Conversely, the "PUT" routine transfers this area from P$ to a specified section of the sccreen. The two routines together allow *PixMix* to "mix up the pieces" of the screen, and enable you to pick up screen pieces and move them around with your joystick. See programs 3 and 4.

Pressing the space bar while solving a puzzle lets you see how the completed puzzle should look. This feature involves the "STRING/SCREEN SWAP" routine. Essentially, this routine simply exchanges the data saved in SCR$ (the original screen image) with the scrambled image on the screen. When the space bar is released, the swap routine is called again, bringing the scrambled image back to screen, and the completed puzzle image back to SCR$. See program 5.

When you solve the puzzle and press the START key, how does *PixMix* know you've solved the puzzle correctly? Once again the program refers to the original image data stored in SCR$, but this time through the "STRING/SCREEN COMPARE" routine. If the screen contents and the string data do not match, the puzzle has not been solved and location 208 ($DO) is set to one to flag this to the BASIC program. *PixMix* then generates its ominous, disapproving hum. On the other hand, if the

continued on page 162

Control GEM
continued from page 158

at the opposite corner, and ptsin + 6 the associated y-value.

The intin variables relate to any integer values passed to the VDI routine. They could be used to indicate the start and end of an arc or of a pie- shaped graphics.

Parameter Matrix

There are other parameters that can be used with VDI, but these will suffice to illustrate some basic ideas. They support the majority of the GDP functions, which we'll now look at in the form of a matrix. The entries here cover only the GDP activities. A more extensive set can be found in the *GEM VDI PROGRAMMER'S GUIDE*. See Figure 7 on page 180.

Note that several entries have an * beside them. Where it appears, the first number is used to obtain an arc, the second to obtain a pie slice. The *2 is used with the ellipse column and implies, if no arc or pie shapes are desired the intin entries can be ignored?

The above entries, in conjunction with the programs provided, should allow you to explore the GDP function of the VDI. This is but a subset of VDI's total capabilities. An additional example, outside the GDP function, is given for drawing lines. It illustrates that the basic parameters still apply, but merely have a different connotation.

And Now To Application Environment Services (AES)

In using the AES, we go to an address for our control, identified as gb in the program. The address for the global array is accessed at gb + 4, and in and out entries as reflected in the program. The address references specify sets of pointers used by the system during processing of this program. And, of course, the x-, y-coordinates are found in the address contained in gintout + 2 and gintout + 4. Once our initial values are set up the routine is recycled. This is done by a call to parameter 79, which is the control parameter for obtaining the mouse state. As we move our mouse, we repetitively generate a new set of x-, y-coordinates, and then peek at them. The process here is a little slow, so move your mouse slowly. To get the displayed point to line up with the mouse indicator's point, we had to subtract 22 from the y-coordinate. See program 7 on page 180.

It seems an empty gesture to develop such a program and not demonstrate its qualities. Therefore I have exercised my hidden artistic talent and drawn a self-portrait that would grace the walls of any museum. No snide remarks, please! See figure 8 on page 180.

Summary

The GEM VDI capabilities can be accessed through the BASIC instructions. They provide an extensive set of functions not always available otherwise. They use a pattern of parameters that establish the resulting product. For drawing purposes, a subset, the GDP, is readily usable. The provided matrix covers the essential entries for GDP activity. •

Real Estate
continued from page 72

real estate/mortgage banking services. Each analysis describes a property type, program structure, hardware and operating requirements, documentation, installation, technical and user support, price and terms, and program availability.

In FIGURE 1 we have presented a bibliography of relevant books in print. FIGURE 2 is a bibliography of relevant magazine articles. Both of these lists were acquired from an online bibliographic database available through a service called DIALOG Information Services, Inc. (Such searches may be expensive or inexpensive depending on the database accessed and the information required. In this case, the cost was approximately $150.00)

Using data which is available online from the comfort of the chair in front of one's personal computer is a very underutilized facility. FIGURE 3 is a list of online data services which have been categorized as real estate industry sources of information by "Data Bases for Business: Profiles and Applications" (Van Mayros and D. Michael Werner; Chilton Book Company, Radnor, PA, 1982). The sources are diverse and may or may not be relevant to any but a small sector of the industry.

Others are also available which are not listed here. For example, Matthew Lesko lists in "The Computer Data and Database Source Book" (Avon Books, New York, 1984), the "Real Estate Information System Network". The subject of this online source is U.S. commercial and industrial real estate. The source of data is real estate agents and brokers. The database contains change property data from throughout the U.S. It covers larger commercial and industrial properties with data on size, type, building age, number of buildings, number of stories, land mass, rental area, price at sale and rental, net operating costs and various other characteristics of the property. All sold properties are included. The producer is the National Association of Realtors, 430 N. Michigan, Chicago, IL 60611 (313-329-8200). The information is available on the Real Estate Information System Network at an approximate cost of $50.00 to cover a one-time fee, $10.00 monthly, and $11.00 per connect hour plus an additional minimal fee.

There are many software guides available at the book and magazine stores. Keep in

continued on page 163

Applying The Atari
continued from page 160

screen does match the string, the flag is left at zero and the congratulatory tune is played by the BASIC program. See program 6.

These five routines are the vital organs of *PixMix*. BASIC serves as glue to hold it all together. *PixMix* serves as a good example of the power and flexibility of the BASIC/machine language combination.

Operating System Keypad Driver

Ever since the CX-85 keypad drivers for BASIC programming were published in the July and August 1985 issues, I've been unundated with requests for drivers to work with just about every commercial software product imaginable. Unfortunately, finding a place in memory for the kaypad driver is not easy; each commercial program would have to be looked at as an individual case. But XL and 130XE owners will be happy to know that the extra RAM in your machines ("underneath" the OS ROM") is the ideal place for such a keypad driver. This is precisel where we install this month's keypad driver.

Type in the *"CX-85 Driver Creator"* using *Program Perfect* and SAVE the program to diskette. If you're not using *Program perfect*, recheck your typing several times, since one small mistake in the data statements can blow the whole program. See program 7.

Now place a formatted diskette (containing DOS files) into the drive and RUN the program. You'll be instructed to press the START key to write an AUTORUN.SYS file — when you're ready, press it.

When the "COMPLETED" message is printed, you're finished. You now have keypad driver that takes up none of your Atari's normal memory and will work with some commercial software.

Using The New Driver

The best way to test out the new driver is by booting up a BASIC disk. Turn off your computer and insert the keypad driver diskette into the disk drive. Now turn on the computer. If everything is O.K. you'll soon get an "INSERT PROGRAM DISKETTE AND PRESS SELECT" prompt. Replace the keypad driver with a regular BASIC diskette and press SELECT. The computer will reboot, and when finished, your Atari will work normally with one exception: It will now respond to a CX-85 keypad plugged into joystick **port 2.** Joystick port 2 was chosen instead of port 1 since many readers wrote that they like to keep a joystick plugged into port 1.

Using Commercial Software

To use the keypad with a commercial product, boot up the keypad driver diskette as described before and remove the diskette. Now place the program you want to load into the drive. If BASIC must be disabled, then hold the OPTION key down while you press SELECT. Otherwise, just press SELECT. The Atari will simulate being turned off and then on and the program should load.

In most instances, you should be able to use the keypad with a commercial software product, but there are exceptions — namely, some heavily protected software and programs that read the Atari's keyboard register directly, bypassing location 764. In these cases, the program will either not run, or will not respond to the keypad.

I'm asking readers who try this keypad driver with commercial programs to please write and let me know which of your programs are compatible with the driver and which aren't (you could even jot the names down on the postcard if you're entering our drawing!) I'll report the findings in a future column.

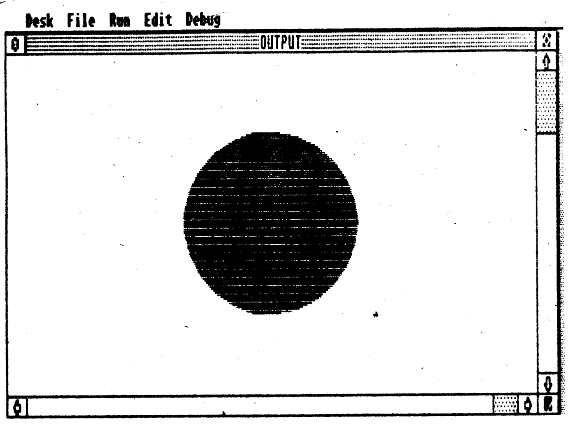
How It Works

When the keypad driver is loaded into an XL or XE, it transfers the Atari's ROM operating system (OS) into the RAM that lies underneath it. Next, the actual keypad

continued on page 188

Programs for Control GEM

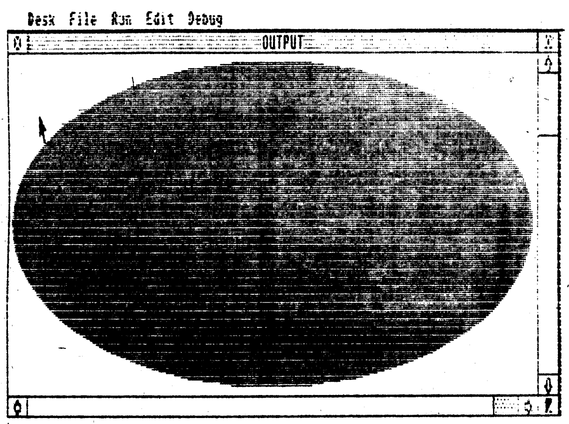
CIRCLE PROGRAM




```
10      ' draw a circle at 304,105 with radius 100
20      color 1,3,2,1,1:fullw 2
30      poke contrl,11
40      poke contrl+2,3
50      poke contrl+6,0
60      poke contrl+10,4
70      poke ptsin,304
80      poke ptsin+2,105
90      poke ptsin+8,100
100     vdisys(1)
```

FIG 1

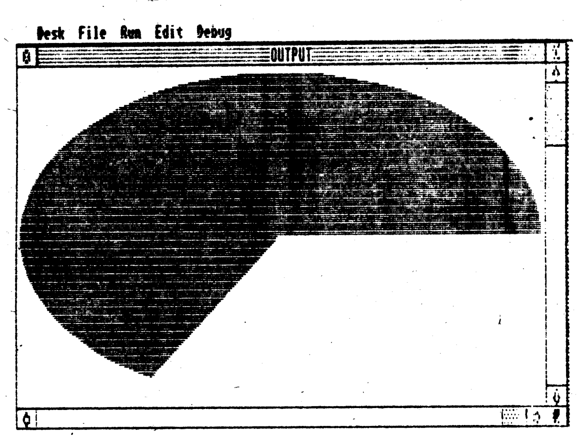
ELLIPSE PROGRAM




```
10      ' draw an ellipse at 304,105  with radius 300,80
20      color 1,3,2,1,1:fullw 2
30      poke contrl,11
40      poke contrl+2,2
50      poke contrl+6,0
60      poke contrl+10,5
70      poke ptsin,304
80      poke ptsin+2,105
90      poke ptsin+4,300
100     poke ptsin+6,80
110     vdisys(1)
120     goto 120
```

FIG 2

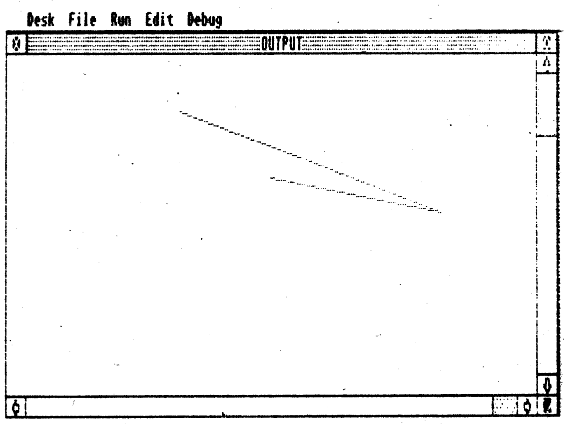
ELLIPTICAL PIE PROGRAM




```
10      ' draw an elliptical pie centered at 304,105
15      clearw 2
20      color 1,3,2,1,1:fullw 2
30      poke contrl,11
40      poke contrl+2,2
50      poke contrl+6,2
60      poke contrl+10,7
70      poke ptsin,304
80      poke ptsin+2,105
90      poke ptsin+4,300
100     poke ptsin+6,80
102     poke intin, 0
104     poke intin+2,2400
110     vdisys(1)
120     goto 120
```

FIG 3

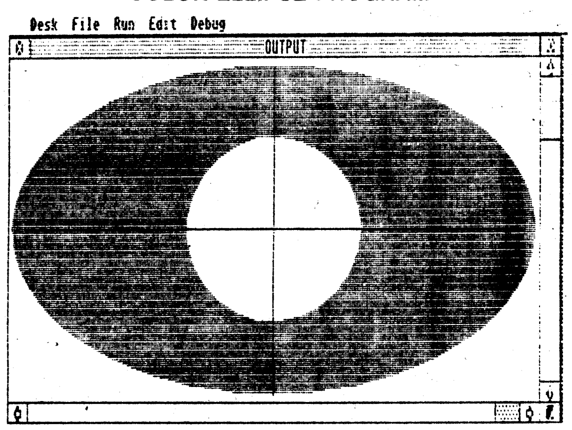
POLLYLINE PROGRAM




```
10      ' draw a couple of lines using polyline method
20      color 1,3,2,1,1:fullw 2
30      poke contrl,6
40      poke contrl+2,3
50      poke contrl+6,0
60      poke ptsin,304
70      poke ptsin+2,83
80      poke ptsin+4,500
90      poke ptsin+6,100
100     poke ptsin+8,200
110     poke ptsin+10,50
120     vdisys(1)
```

FIG 4

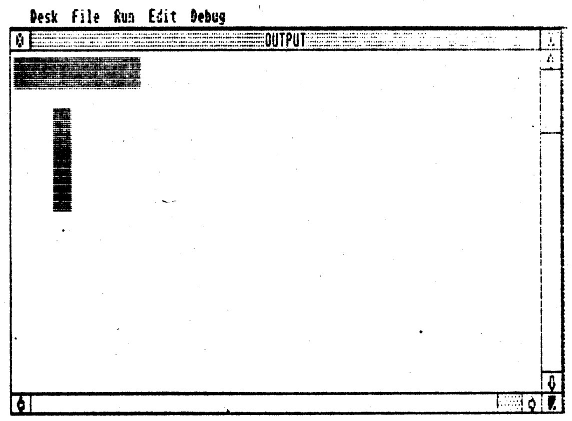
COLOR ELLIPSE PROGRAM




```
20      clearw 2
30      ' draw an ellipse at 304,105  with radius 300,80
40      color 1,2,2,1,1:fullw 2
50      poke contrl,11
60      poke contrl+2,2
70      poke contrl+6,0
80      poke contrl+10,5
90      poke ptsin,304
100     poke ptsin+2,105
110     poke ptsin+4,300
120     poke ptsin+6,80
130     vdisys(1)
140     ' draw a circle at 304,105 with radius 100
150     color 1,1,2,1,1:fullw 2
160     poke contrl,11
170     poke contrl+2,3
180     poke contrl+6,0
190     poke ptsin+8,304
200     poke contrl+10,4
210     poke ptsin,304
220     poke ptsin+2,105
230     poke ptsin+8,100
240     vdisys(1)
250     ' draw X-Y axes using polyline method
260     color 1,1,1,1,1:fullw 2
270     poke contrl,6
280     poke contrl+2,5
290     poke contrl+6,0
300     poke ptsin, 25
310     poke ptsin+2,105
320     poke ptsin+4,575
330     poke ptsin+6,105
340     poke ptsin+8,304
350     poke ptsin+10,105
360     poke ptsin+12,304
370     poke ptsin+14,25
380     poke ptsin+16,304
390     poke ptsin+18,185
400     vdisys(1)
410     goto 410
```

FIG 5

BAR GENERATION PROGRAM




```
10      ' draw a bar from 5,25  across to 150,40
20      color 1,2,2,1,1:fullw 2
30      poke contrl,11
40      poke contrl+2,2
50      poke contrl+6,0
60      poke contrl+10,1
70      poke ptsin,5
80      poke ptsin+2,25
90      poke ptsin+4,150
100     poke ptsin+6,40
110     vdisys(1)
120     ' draw a bar from 50,100 up to 50,50
130     poke contrl,11
140     poke contrl+2,2
150     poke contrl+6,0
160     poke contrl+10,1
170     poke ptsin,50
180     poke ptsin+2,100
190     poke ptsin+4,70
200     poke ptsin+6,50
210     vdisys(1)
220     goto 220
```

FIG 6

MOUSE POSITION PROGRAM

	BAR	ARC	CIRCLE	ELLIPSE	RECTANGLE
Contrl	11	11	11	11	11
Contrl+2	1	4	3	2	2
Contrl+6	0	2	0	0 (2)*	0
Contrl+10	1	2 (3)*	4	5 (6) (7)*	8 (9)*
Ptsin	X(1)	X(center)	X(center)	X(center)	X(1)
Ptsin+2	Y(1)	Y(center)	Y(center)	Y(center)	Y(1)
Ptsin+4	X(2)	0	0	X-radius	X(2)
Ptsin+6	Y(2)	0	0	Y-radius	Y(2)
Ptsin+8	----	0	radius	----	----
Ptsin+10	----	0	0	----	----
Intin	----	start <	----	start < *2	----
Intin+2	----	end <	----	end <	----

```
10      ' print mouse x,y position
12      clearw 2
20      a#=gb
30      control=peek(a#)
40      global=peek(a#+4)
50      gintin=peek(A#+8)
60      gintout=peek(a#+12)
70      addrin=peek(a#+16)
80      addrout=peek(a#+20)
85      AGAIN:
90      gemsys(79)
100     ax = peek(gintout+2)
102     ay = peek(gintout+4)
105     ' print ax
110     ' print ay
140     if peek(gintout+6)<>0 then linef ax,ay-22,ax,ay-22
200     goto AGAIN
```

FIG 7

PORTRAIT OF AUTHOR AS BEGINNING GRAPHICS PROGRAMMER

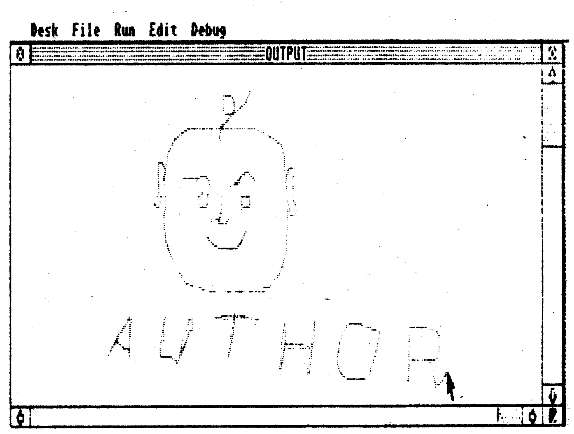


FIG 8

Applying The Atari continued from page 162

decoding routine is loaded into the Atari's international character set memory at $CC00 to $CCD3. The Atari's actual vertical blank interrupt routine is then rerouted to execute the keypad routine. This is the beauty of having a RAM OS — we can add extensions to the *actual* Atari OS without having to worry about vectors that might be utilized or changed by the next program loaded in. None of the application RAM is used, and thus the keypad driver is transparent to most programs.

The next step is to display the introductory message on the screen and wait for the SELECT key to be pressed. When it's pressed, a jump is made to the Atari's cold start vector (COLDSV, $E477) so that the next diskette can be booted. Necessary portions of the OS are changed before jumping to COLDSV to prevent the ROM OS from being reenabled and to prevent the COLDSV routine from clearing the RAM OS when the rest of the memory is zeroed.

Next Month

We'll take a look at the assembly listing for the kaypad routine, plus we'll enter a personal budget program, designed in response to readers' requests.

Readers' questions, comments and contributions are welcome. Please enclose a self-addressed, stamped envelope (SASE) for a personal reply.

A diskette of the programs listed in this month's column is available from the author for $7.00, postpaid. Please make checks payable to "Jeff Brenner" and specify your disk drive model.

"Program Perfect" is a utility used to check for typing errors while entering programs from this column. Readers may send $5.00 for a diskette or a SASE for a listing of this program. Address all correspondence to:

Jeff Brenner
"Applying The Atari 4/86"
c/o *Computer Shopper*
PO Box F
Titusville, FL 32781-9990

Improved Directory Program

```
1 REM DIRECTORY PROGRAM MOD. BY N. WORTH
2 DIM FILE$(17),R$(1)
3 OPEN #1,6,0,"D:*.*"
4 INPUT #1,FILE$
5 PRINT FILE$;CHR$(32);CHR$(32);
6 IF FILE$(5,8)="FREE" THEN 8
7 GOTO 4
8 CLOSE #1
9 ? :? :? "RUN AGAIN (Y/N)";:INPUT R$
10 IF R$="Y" THEN ? :? "INSERT DISK AND PRESS RETURN";:INPUT
#16;R$
11 IF R$="N" THEN ? CHR$(125):NEW
12 GOTO 3
```

PROGRAM 1

STRING/SCREEN SWAP ROUTINE

```
10  *=$600 ; RELOCATABLE
20 ; STRING/SCREEN SWAP
30 ; SWAPS 7680 BYTES OF STRING W/GR.8 SCREEN
40 ; FORM: A=USR(ADR(SWAP$),ADR(A$))
50 ; WHERE SWAP$ IS THIS ROUTINE       36 ; AND A$ IS STRING TO SWAP
60  PLA
70  PLA
80  STA $CD  ; SAVE STRING ADR
90  PLA
0100  STA $CC
0110  LDA 88
0120  STA $CE ; SAVE SCREEN ADR
0130  LDA 89
0140  STA $CF
0150  LDX #192 ; 192 LINES
0160 LOOP2 LDY #39 ; 40 BYTES/LINE
0170 LOOP LDA ($CC),Y ; GET BYTE OF STRING
0180  PHA              ; SAVE IT
0190  LDA ($CE),Y   ; GET BYTE OF SCREEN DATA
0200  STA ($CC),Y   ; STORE IN STRING
0210  PLA              ; RETRIEVE STRING BYTE
0220  STA ($CE),Y   ; STORE IN SCREEN MEM
0230  DEY
0240  BPL LOOP
0250  CLC
0260  LDA $CC
0270  ADC #40   ; NEXT STRING "LINE"
0280  STA $CC
0290  LDA $CD
0300  ADC #0
0310  STA $CD
0320  CLC
0330  LDA $CE
0340  ADC #40   ; NEXT SCREEN LINE
0350  STA $CE
0360  LDA $CF
0370  ADC #0
0380  STA $CF
0390  DEX
0400  BNE LOOP2
0410  RTS
0420  .END
```

PROGRAM 5

PUT SCREEN PIECE ROUTINE

```
10  *=$600  ; RELOCATABLE
20 ; PUT BLOCK ROUTINE
30 ; (c)1986 JEFF BRENNER
40 ; TRANSFER STRING TO SECTION OF GR.8 SCREEN
50 ; (8 BYTE BY 32 LINE SECTION)
60 ; FORM A=USR(PUT,ADR(P$),SCREEN)
70 ; DIM P$(320), P$ STORES DATA
80 ; SCREEN=SCREEN LOC. TO START PUT
90  PLA
0100  PLA
0110  STA $CD  ; SAVE STRING ADR
0120  PLA
0130  STA $CC
0140  PLA
0150  STA $CF  ; SAVE SCREEN ADR
0160  PLA
0170  STA $CE
0180  LDX #32  ; VERTICAL BYTES (LINES) FOR PIECE
0190 AGAIN LDY #7  ; * HORIZONTAL BYTES FOR PIECE-1
0200 LOOP1 LDA ($CC),Y
0210  STA ($CE),Y
0220  DEY
0230  BPL LOOP1
0240  CLC
0250  LDA $CE
0260  ADC #40     ; NEXT SCREEN LINE
0270  STA $CE
0280  LDA $CF
0290  ADC #0
0300  STA $CF
0310  LDA $CC
0320  ADC #8      ; * HORIZONTAL BYTES FOR PIECE
0330  STA $CC
0340  LDA $CD
0350  ADC #0
0360  STA $CD
0370  DEX
0380  BNE AGAIN
0390  RTS
0400  .END
```

PROGRAM 4

GET SCREEN PIECE ROUTINE

```
10  *=$600 ; RELOCATABLE
20 ; GET BLOCK ROUTINE
30 ; (c)1986 JEFF BRENNER
40 ; TRANSFER SECTION OF GR.8 SCREEN TO STRING
50 ; (FOR 8 BYTE BY 32 LINE SECTION)
60 ; FORM A=USR(GET,ADR(P$),SCREEN)
70 ; DIM P$(320), P$ STORES DATA
80 ; SCREEN=SCREEN LOC. TO START GET
90  PLA
0100  PLA
0110  STA $CD  ; SAVE STRING ADR
0120  PLA
0130  STA $CC
0140  PLA
0150  STA $CF  ; SAVE SCREEN ADR
0160  PLA
0170  STA $CE
0180  LDX #32  ; VERTICAL BYTES (LINES) FOR PIECE
0190 AGAIN LDY #7  ; * HORIZ BYTES FOR PIECE-1
0200 LOOP1 LDA ($CE),Y ; GET BYTE FROM SCREEN
0210  STA ($CC),Y ; STORE IN STRING
0220  LDA #0
0230  STA ($CE),Y ; ERASE BYTE OF SCREEN
0240  DEY
0250  BPL LOOP1
0260  CLC
0270  LDA $CE
0280  ADC #40      ; NEXT SCREEN LINE
0290  STA $CE
0300  LDA $CF
0310  ADC #0
0320  STA $CF
0330  LDA $CC
0340  ADC #8       ; * HORIZ BYTES PER PIECE
0350  STA $CC
0360  LDA $CD
0370  ADC #0
0380  STA $CD
0390  DEX
0400  BNE AGAIN
0410  RTS
0420  .END
```

PROGRAM 3

CX-85 DRIVER CREATOR

```
HTJ 10 REM CX-85 NUMERIC KEYPAD DRIVER CREATOR
KGJ 20 REM COPYRIGHT 1986 JEFF BRENNER
TPJ 30 PRINT CHR$(125);"INSERT FORMATTED DISKETTE INTO DRIVE"
PNJ 40 PRINT :PRINT "PRESS <START> TO WRITE AUTORUN FILE."
LQJ 50 IF PEEK(53279)<>6 THEN 50
FAJ 60 PRINT "CREATING AUTORUN.SYS...":OPEN #3,8,0,"D:AUTORUN.SYS"
TFJ 70 FOR I=1 TO 402:READ N:PUT #3,N:NEXT I
YOJ 80 CLOSE #3
MUJ 90 PRINT "COMPLETED.":END
QAJ 100 DATA 255,255,0,6,167,6,120,169,0,141,14,212
RKJ 110 DATA 174,1,211,160,0,185,0,192,206,1,211,153
TOJ 120 DATA 0,192,142,1,211,200,208,241,238,13,6,238
UHJ 130 DATA 19,6,173,13,6,240,14,201,208,208,226,169
RUJ 140 DATA 216,141,13,6,141,19,6,208,216,206,1,211
SFJ 150 DATA 88,96,169,0,141,14,212,162,5,141,10,212
DRJ 160 DATA 202,208,253,169,76,141,138,194,169,0,141,139
JTJ 170 DATA 194,169,204,141,140,194,169,146,141,133,196,169
YTJ 180 DATA 6,141,134,196,160,9,185,85,204,153,220,194
EDJ 190 DATA 136,16,247,160,26,185,94,204,153,169,196,136
BHJ 200 DATA 16,247,169,64,141,14,212,160,79,185,132,204
UUJ 210 DATA 56,233,32,9,128,145,88,136,16,243,173,31
WGJ 220 DATA 208,41,2,208,249,76,119,228,173,1,211,141
DJJ 230 DATA 255,207,32,218,196,169,254,141,1,211,173,255
ICJ 240 DATA 207,141,1,211,96,0,226,2,227,2,0,6
NZJ 250 DATA 224,2,225,2,56,6,0,204,211,204,174,17
VUJ 260 DATA 208,240,6,202,142,82,204,240,47,174,121,2
WVJ 270 DATA 236,84,204,208,5,173,82,204,208,34,173,83
XYJ 280 DATA 204,240,5,206,83,204,240,24,238,82,204,142
RYJ 290 DATA 84,204,238,83,204,173,3,210,201,1,240,2
AXJ 300 DATA 162,16,189,64,204,141,252,2,104,168,104,170
MTJ 310 DATA 104,64,52,24,29,27,35,51,53,48,43,31
IDJ 320 DATA 30,26,50,34,12,14,28,155,0,0,0,169
QHJ 330 DATA 4,133,1,169,0,76,121,204,169,160,133,6
TQJ 340 DATA 173,0,160,73,255,141,0,160,205,0,160,208
SZJ 350 DATA 9,73,255,141,0,160,169,192,133,6,96,160
MZJ 360 DATA 0,153,0,6,200,208,250,76,3,195,67,88
MNJ 370 DATA 45,56,53,32,75,69,89,80,65,68,32,68
LYJ 380 DATA 82,73,86,69,82,32,45,32,40,67,41,49
MQJ 390 DATA 57,56,54,32,74,46,32,66,82,69,78,78
MPJ 400 DATA 69,82,73,78,83,69,82,84,32,80,82,79
MNJ 410 DATA 71,82,65,77,32,68,73,83,75,69,84,84
MFJ 420 DATA 69,32,65,78,68,32,80,82,69,83,83,32
DUZ 430 DATA 83,69,76,69,67,84
```

PROGRAM 7

STRING TO SCREEN ROUTINE

```
10  *=$600 ; RELOCATABLE
20 ; STRING TO GR.8 SCREEN
30 ; (C)1986 JEFF BRENNER
40 ; FORM: A=USR(ADR(STS$),ADR(A$))
50  PLA
60  PLA
70  STA $CD   ; SAVE STRING ADR
80  PLA
90  STA $CC
0100  LDA 88
0110  STA $CE ; SAVE SCREEN ADR
0120  LDA 89
0130  STA $CF
0140  LDX #192   ; 192 LINES
0150 LOOP2 LDY #39 ; 40 BYTES/LINE
0160 LOOP LDA ($CC),Y
0170  STA ($CE),Y
0180  DEY
0190  BPL LOOP
0200  CLC
0210  LDA $CC
0220  ADC #40
0230  STA $CC
0240  LDA $CD
0250  ADC #0
0260  STA $CD
0270  CLC
0280  LDA $CE
0290  ADC #40
0300  STA $CE
0310  LDA $CF
0320  ADC #0
0330  STA $CF
0340  DEX
0350  BNE LOOP2
0360  RTS
0370  .END
```

PROGRAM 2

STRING/SCREEN COMPARE ROUTINE

```
10  *=$600 ; RELOCATABLE
20 ; COMPARE STRING W/GR.8 SCREEN
30 ; (C)1986 JEFF BRENNER
40 ; FORM: A=USR(ADR(SSC$),ADR(A$))
50 ; WHERE SSC$ IS THIS ROUTINE
60 ; AND A$ IS STRING TO COMPARE
70 ; RESULT: IF PEEK(208)>0, NO MATCH
80  PLA
90  PLA
0100  STA $CD
0110  PLA
0120  STA $CC
0130  LDA #0
0140  STA $D0 ; CLEAR $D0
0150  LDA 88
0160  STA $CE ; SAVE SCREEN ADR
0170  LDA 89
0180  STA $CF
0190  LDX #192 ; 192 LINES
0200 LOOP2 LDY #39  ; 40 BYTES/LINE
0210 LOOP LDA($CC),Y ;GET STRING VALUE
0220 CMP ($CE),Y ;COMP. W/SCREEN VALUE
0230  BNE END ; IF NOT EQUAL, END
0240  DEY
0250  BPL LOOP
0260  CLC
0270  LDA $CC
0280  ADC #40 ; NEXT STRING "LINE"
0290  STA $CC
0300  LDA $CD
0310  ADC #0
0320  STA $CD
0330  CLC
0340  LDA $CE
0350  ADC #40 ; NEXT SCREEN LINE
0360  STA $CE
0370  LDA $CF
0380  ADC #0
0390  STA $CF
0400  DEX
0410  BNE LOOP2
0420  RTS
0430 END INC $D0 ; SET $D0 TO 1
0440  RTS
0450  .END
```

PROGRAM 6

That Dangerous Thing
continued from page 143

cue. This is easily modified by changing those lines involving Open and Field.

Analysis Of Tymon's Tutor

The program is rather long, but it is divided into a sequence of logical modules, simplifying both programming and understanding. There are several modules which exercise the sound capability of the Jackintosh. These are represented by the module, INITMUSIC. (By the way, each module is identified by having its title in capital letters.)

In INITMUSIC, as well as the other routines using sound -- INMUSIC, LOSEMUSIC, WINMUSIC -- the first line establishes the characteristics of the wave. These entries can be confusing, as the Atari ST BASIC Sourcebook is a little unclear on their use. Note first that our inputs are in decimal form. Each numeric value is the equivalent of corresponding binary values. For example, in instruction Wave 7, 5, 15, 10, 25, the intiial 7 is equivalent to 00000111. On my machine this enables voices 1, 2 and 3, and precludes noise generation. This is exactly opposite the Sourcebook guidelines. The entries for envelope (5), and for shape (15) operate in the same manner. The latter two entries, 10 and 25, reflect the period and duration of the envelope in number of 1/50th second increments.

Under INITIALIZE the learning package file is identified, and a relative file (called in the Sourcebook and elsewhere, erroneously, a random file) is opened and the fields prescribed. SHOWME then permits the user to examine all the records on the learning package, inputting entries to the learning package, or correcting existing entries in the learning package. Note that, in using the relative files, entries are made first to a buffer record, using the LSET instruction. This is followed by outputting to the buffer with the PUT instruction. Similarly, the record is read from the disk to a buffer, using the GET instruction. The field contents can then be used. Field contents cannot be numeric entries. Before these can be used they must be converted to strings.

Prompts lead the user in placing entries on the disk. In general he will enter a 1 (indicating initial entry), the entry to be learned, and the entry to be displayed as a cue to the user. The process is repeated until the user enters the sequence: last,a,

Similarly, on using the package in a learning mode, the data is extracted from the disk, and the cue is displayed on the screen in red. The user is prompted to enter the appropriate response. If he fails to do so the correct information is displayed on the screen, at the same time the sound generator plays a sympathetic theme. If the entry is correct the system congratulates him, and the sound generator outputs a triumphant march. Eh, anyway, some noise. The record has its key changed to the next higher value. Thus, unless the user requests a key value of higher than 1, he will review only 1-coded records. He may ask for keys up to 6 (corresponding, in my use, to one year, where daily review is key 1, weekly is key 2, etc.) The learning mode is repeated until there are no more records to be read, or the user enters LAST. This activity is accomplished under the module READBACK.

While in the correction mode the module FIXIT dominates the activity, and permits either the required response field, or the display cue field, to be modified. It allows subsequent records to also be corrected. However, this is possible only for records which follow the current one on the disk. That is, the records are actually sequential. The access is relative. To actually be able to randomly select records we would need a second file which would contain a key for selection, and a relative record number. We could then find the key in this file, and read the relative record associated with it. Our current program doesn't provide this, although it is a simple exercise to develop the capability.

This has been a thumbnail sketch of the program. By reviewing the code, running the program (both in learning package development and use), and analyzing the results you'll find that the process is straightforward.

Summary

The program permits you to tailor a sequence of learning sessions to your personal needs. Need to learn a specialized vocabulary? Using this program you can rapidly generate a file which can then be used for review -- but which only reviews those words you don't know. Unless, of course, you want to review all or a subset. Do you need to learn the capitols of the states? The same principle applies. Need to learn a foreign vocabulary? Again, you can develop a tailored program. You can develop a program to help you learn concepts, words, formulas, or any other information which can be entered into the system.

The program is designed to cover a variety of situations. Nevertheless occasions may arise where it isn't entirely satisfactory. The program is written in BASIC, and is widely commented. As a result, if needed, you may make changes which will increase the value of the package.

A commercial version of the package will soon be available. For readers who aren't too eager to type in the code there is available a package consisting of a 3½" disk containing the program and an extensive English vocabulary learning package. This is accompanied by a complete set of instructions, a list of the English vocabularly file (other learning packages are being generated), and a list of the tutor.bas program. The package is available for $25.00 from: Frank Tymon, 4749 W. K-12 Ave., Lancaster, CA 93536. ●

Program continued from page 192

```
1910  sound 2,0,0,0,0
1920  sound 3,0,0,0,0
1930  return
1940
'******************************************************
1950  CONGRAT:
1960  gotoxy 5,5
1970  color 2:? "CORRECT--CONGRATULATIONS"
1980  return
1990
'******************************************************
2000  WINMUSIC:
2010  wave 7,0,0,0,20
2020  for j=1 to 3
2030  for i=1 to 3
2040  sound 1,15,3,3,1*i
2050  sound 1,5,3,3,1*i
2060  next i
2070  for k=1 to 100:next k
2080  next j
2090  sound 1,0,0,0,0
2100  color 1
2110  return
2120
'*************************************************
2130  SORRY:
2140  gotoxy 5,2
2150  color 2:? "WRONG--SORRY"
2160  return
2170
'*************************************************
2180  LOSEMUSIC:
2190  wave 7,0,0,0,25
2200  for ab=1 to 3
2210  for i=1 to 3
2220  for j=1 to 3
2230  sound 1,5*i*j,5-j,1*i*j
2240  sound 1,15/i*j,1+j,1*i*j*5
2250  next j:next i
2260  sound 1,0,0,0,0
2270  next ab
2280  color 1
2290  return
2300
'**************************************************
2310  FIXIT:
2320  nm=lof(1)
2330  NEXTDEL:
2340  color 2
2350  ? "input the required response field to locate record"
2360  ? "the required response field is entry you ordinarily
will return"
2370  ? "in response to the displayed cue."
2380  input "input the entry here: ",rr$
2390  for i=1 to nm
2400  get 1,i
2410  tag$=wd$:k$=key$:date$=dt$:mn$=meaning$
2420  if (instr(0,tag$,rr$)=0) then goto SHLOGON
2430  color 3
2440  print k$;" ";date$;" ";tag$;" ";mn$
2450  color 2
2460  ADJUST:
2470  ? "input 1,2,3, or 4 to go to next record, discontinue
changes,"
2480  input "change required response field, or change
meaning field ";asel
2490  if asel=1 then goto NEXTDEL
2500  if asel=2 then close 1:END
2510  if asel=3 then goto DELWORD
2520  if asel=4 then input "input meaning ";mn$:goto REENTER
2530
'*****************************************************
2540  DELWORD:
2550  input "input word ";tag$
2560
'*****************************************************
2570  REENTER:
2580  lset key$=k$:lset dt$=d$:lset wd$=tag$:lset
meaning$=mn$
2590  put 1,i
2600  goto NEXTDEL
2610
'******************************************************

2620  SHLOGON:
2630  next i
2640  close 1
2650  color 1
2660  END
2670
'******************************************************
**
2680  CLEARIT:
2690  for i=0 to 3
2700  fullw i:clearw i
2710  next i
2720  RETURN
2730
'******************************************************
**
3020  SHOWME:
3030  clearw 2
3040  For qu=1 to lof(1)
3050  get 1,qu
3060  color 3
3070  print key$,dt$,wd$,meaning$
3080  next qu
3090  gosub PAUSE
3100  RETURN
3110  TRUNCATE:
3120  n=15
3130  b$=right$(buff$,1)
3140  if (b$=" ") then n=n-1:buff$=left$(buff$,n):goto 3130
3150  RETURN
```